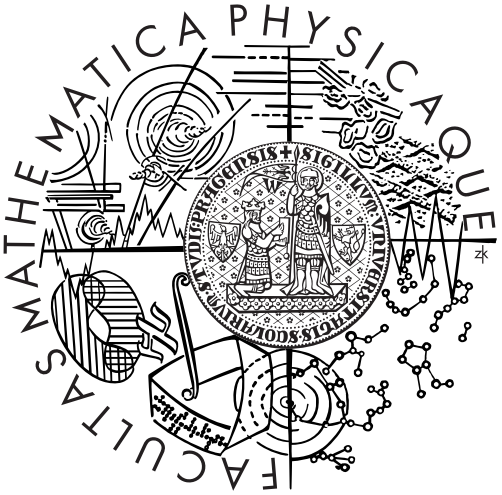
Charles University in Prague
Faculty of Mathematics and Physics

MASTER THESIS



Jakub Marek

Exploiting GPS in Monte Carlo Localization

Department of Software Engineering

Supervisor of the master thesis: RNDr. David Obdržálek, Ph.D.

Study programme: Computer science
Specialization: Software systems

Prague 2013

I would like to thank to the supervisor of this thesis, RNDr. David Obdržálek, Ph.D. for his guidance and advice on this thesis.

I would also like to thank to my family and friends for the patience (mostly) and support during the long time it took to finish this work and to all the people around me who didn't run off screaming every time I started babbling about pseudoranges.

The team of Roboauto Karlík let me record real life data from their robot at the Robotour competition and Tomáš Ondráček helped me with parsing of Karlík's log files. Thank You.

My eternal gratitude goes to my mom and Marika for language corrections during the last few days before submission date. It is because of them that sentences in this text mostly have beginning, middle and an end.

And finally thanks to my lazy self, for not backing out this time.

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In Prague, April 12, 2013 Jakub Marek

Contents

1 Introduction **1**

2 GPS **3**

2.1 Basic Description . . . 3
2.2 Operation Principles . . . 4
2.2.1 Time of Arrival Measurements . . . 4
2.2.2 Time and GPS . . . 4
2.2.3 GPS Signals . . . 5
2.2.4 Navigation Messages . . . 5
2.2.5 Reference Frames . . . 6
2.2.6 Ephemeris . . . 7
2.3 Obtaining Position and Velocity . . . 8
2.3.1 Pseudorange . . . 8
2.3.2 Position . . . 9
2.3.3 Velocity . . . 9
2.4 Measurement Errors . . . 10
2.4.1 User Equivalent Range Error . . . 10
2.4.2 Dilution of Precision . . . 11
2.4.3 Error Sources . . . 12
2.5 Methods for Improving GPS Precision . . . 15
2.5.1 Kalman Filters . . . 15
2.5.2 Differential GPS . . . 16
2.5.3 Assisted GPS . . . 16
2.6 Protocols . . . 16
2.6.1 NMEA 0183 . . . 17
2.6.2 RINEX . . . 17
2.6.3 Proprietary Protocols . . . 17
2.7 Similar Systems . . . 17
2.7.1 Galileo . . . 18

3 Monte Carlo Localization **19**
3.1 Localization . . . 19
3.2 Markov Localization . . . 19
3.2.1 Assumptions . . . 20
3.2.2 Derivation . . . 20
3.2.3 Algorithm . . . 21
3.2.4 Density Representations . . . 21
3.2.5 Kalman Filter . . . 21
3.3 Monte Carlo Localization . . . 22
3.3.1 Algorithm . . . 23
3.3.2 Modifications . . . 26

4 GPS Used in Monte Carlo Localization **27**
4.1 Position Domain Integration . . . 27
4.1.1 Robot State . . . 28
4.1.2 Sensor Model . . . 28
4.1.3 Correlation of Position Errors . . . 30
4.2 Measurement Domain Integration . . . 30
4.2.1 Robot State . . . 31
4.2.2 Preprocessing . . . 31
4.2.3 Motion Model . . . 31
4.2.4 Sensor Model . . . 32
4.2.5 Initialization . . . 36
4.2.6 Algorithm . . . 36

5 Implementation **41**
5.1 SiRF III . . . 41
5.1.1 Protocol . . . 41
5.1.2 Measurements . . . 42
5.1.3 Ephemeris . . . 42
5.1.4 Carrier Frequency . . . 42
5.1.5 Clock Jumps in SiRF Output . . . 42
5.2 Design . . . 43
5.2.1 Communication With SiRF Chip . . . 44
5.2.2 Recordings . . . 44
5.2.3 Interfaces to GPS Data . . . 45
5.2.4 Ephemeris . . . 45
5.3 Position Domain Error Model . . . 45
5.3.1 Mapping Coordinate Frames . . . 46
5.4 Measurement Domain Error Model . . . 46

5.4.1 Clock Offset Estimation 46
5.4.2 Applied Corrections 47

6 Conclusion **49**

Bibliography **51**

A DVD Contents **55**

B Datasets **57**

C tl;dr **59**

Title: Exploiting GPS in Monte Carlo Localization
Author: Jakub Marek
Department / Institute: Department of Software Engineering
Supervisor of the master thesis: RNDr. David Obdržálek, Ph.D., Department of Software Engineering

Abstract: This work presents two approaches for integrating data from a low cost GPS receiver in a Monte Carlo localization algorithm. Firstly, an easily applicable method based on data in the standard NMEA protocol is shown. Secondly, an original algorithm utilizing lower level pseudorange measurements accessed in binary receiver-specific format is presented. In addition, a set of tools for analysis of GPS measurement errors on receivers with SiRF III chipset was implemented.

Keywords: robotics, localization, GPS, Monte Carlo localization, pseudorange, SiRF

Název práce: Použití GPS v Monte Carlo lokalizaci
Autor: Jakub Marek
Katedra / Ústav: Katedra softwarového inženýrství
Vedoucí diplomové práce: RNDr. David Obdržálek, Ph.D., Katedra softwarového inženýrství

Abstrakt: Tato práce uvádí dva postupy pro integraci dat z běžného GPS přijímače v Monte Carlo lokalizaci. Jako první popisuje snadno použitelnou metodu založenou na použití dat přístupných ve standardním protokolu NMEA. Dále obsahuje originální algoritmus používající nízkoúrovňová měření pseudorange, získávaná za pomoci proprietárního protokolu daného přijímače. V rámci práce vznikl balík nástrojů pro analýzu chyb měření GPS přijímačů s chipsetem SiRF III.

Klíčová slova: robotika, lokalizace, GPS, Monte Carlo lokalizace, pseudorange, SiRF

1 Introduction

The ability to determine its position is important for autonomous or semi-autonomous robot not only for completing its task, but also for a basic safe navigation through the environment.

The Global Positioning System (GPS) is a satellite navigation system that provides position, velocity and time information for receivers at any place on Earth with direct satellite visibility. Seeing its use in automotive industry, GPS seems to be a good candidate for localization of robots operating in outdoor areas. The best position measurement precisions available from the Global Positioning System are in the orders of millimeters with survey grade hardware. Consumer grade receivers, however, only offer precision of several meters, which by itself isn't good enough for the robot to navigate safely. A price difference of several orders of magnitude makes the use of consumer GPS receivers the only option for many robot designers, though.

Monte Carlo localization (MCL) is an algorithm that estimates position of a robot based on noisy measurements, possibly from multiple sensors. Monte Carlo localization is a type of Markov localization that represents the position estimation as a set of samples. This estimate is periodically updated in prediction and correction steps based on actions the robot takes and on measured sensor data.

The goal of this work is to acquire as much precision as possible from a single low cost GPS receiver by using it as an input to MCL and to make it possible to use GPS as one of the primary sensors for robot localization.

A relatively straightforward way of integrating the GPS data to any localization algorithm is to use WGS84 geodetic coordinates from the standard NMEA protocol. With this approach, however, Kalman filters, position smoothing and other filters typically employed in consumer grade receivers assume that the receiver is mounted in a car or carried on foot. These assumptions are subsequently used to modify the data, examples include minimum speed threshold or vertical speed limits. Additionally, based on NMEA data, measurement errors can only be characterized very roughly.

A different approach, which is explored in this work, uses raw pseudorange measurements from the GPS receiver. When using pseudorange measurements, localization algorithm can work with motion model which closely matches the real hardware. Each

pseudorange measurement can have an independent error model and further modifications to improve precision can be used.

The rest of the text is organized as follows: Chapter 2 describes the GPS system and its operating principles. Chapter 3 describes Monte Carlo localization. Chapter 4 proposes fusing of GPS data into MCL, and Chapter 5 describes implementation of experiments with GPS and Monte Carlo localization. Chapter 6 concludes and evaluates the work.

2 GPS

The Global Positioning System is a global navigation satellite system (GNSS) developed and operated by the US government. GPS provides three-dimensional position and velocity measurements as well as precise time source for every place on Earth with direct satellite visibility. The GPS is not the only GNSS, in use, but arguably the only one in wide use. Very brief overview of other navigation systems can be found in Section 2.7.

This chapter contains an overview of the Global Positioning System and the basics of its operation. This will be used to develop methods for using GPS measurements in Monte Carlo localization in Chapter 4.

First, principles of GPS are discussed, followed by a description of methods for obtaining receiver position and velocity, error sources of GPS measurements and methods of dealing with these errors. Lastly this chapter deals with communication with consumer grade GPS receivers and also briefly with worldwide positioning systems other than GPS.

Unless other sources are cited, this chapter is based on chapters 2 and 7 of [15].

Specification of GPS user segment interface can be found in [11], describing many parts of GPS system in depth. However since in this work we only interface with the GPS system through a consumer-grade receiver, we don't encounter many low level details of the GPS.

2.1 Basic Description

GPS consists of a constellation of at least 24 satellites (space vehicles, SVs), a control segment and end-user receivers.

The satellites are orbiting in six nearly circular orbits approximately $20\,200$ km above Earth surface. The satellite constellation is still being modernized [19] and new satellites are being launched. Several GPS signals with different properties are available and more will become available with modernized satellites, but generally all these signals can be divided to encrypted military signals (Precise Positioning Service) and unencrypted civilian signals (Standard Positioning Service). The civilian signals are available worldwide without any limitations.

GPS is often viewed as a simple and reliable means of navigation, but especially the unencrypted signals are susceptible to spoofing attacks [36].

The control segment consists of a network of ground facilities and is responsible for monitoring the satellite constellation and for sending commands and data to the satellites.

2.2 Operation Principles

2.2.1 Time of Arrival Measurements

GPS calculates position by measuring the time it takes for a signal emitted from a known position to reach the receiver.

Each GPS satellite transmits a pseudo random sequence (PRN signal), containing timing information. The receiver is able to replicate and match this sequence and therefore is able to determine the transmission time of the signal.

If the position of satellites is known and all clocks in the system are synchronized, the position of the receiver can be calculated as an intersection of at least three spheres centered around the satellites and with radius corresponding to time of flight of the signal.

In reality, receiver clock offset needs to be calculated during localization witch adds a new dimension to position determination.

2.2.2 Time and GPS

One of the functions of the GPS system is a dissemination of precise time. There are three basic time frames appearing in the GPS system.

GPS System Time

GPS System Time is a paper time scale based on atomic clock standards in GPS satellites and on the ground. This time standard is not directly available neither in SVs or in user receivers.

It is specified by a week number – number of Saturday/Sunday midnights since week 0 that started on January, 6th 1980 – and time of week in seconds.

GPS system time is related to UTC. It is a continuous time scale, not adjusted for leap seconds and it is required to be within 1 µs from UTC modulo 1 s.

SV Time

SV Time is a value that satellites transmit in their ranging signals, obtained from the satellite's atomic clock. Although the atomic clock standards are highly stable, the

offset between SV Time and GPS System Time may reach up to 1 ms (equivalent to 300 km of measurement error).

Values of this offset are calculated by the control segment and broadcast by the satellites, or downloaded with precise ephemeris (see Section 2.2.6). Because of this we can assume that this value is known, although not with absolute precision.

Receiver Time

Receiver Time is a time that is kept by the GPS receiver clock.

Receivers are usually equipped only with simple crystal oscillators and aren't capable of keeping precise time. This is solved by adding the clock offset as a fourth dimension of the receiver position. Having clock offset as a part of the navigation solution makes it easy to calculate precise time after a fix is obtained by adding the offset to the receiver clock.

2.2.3 GPS Signals

GPS signals are transmitted using CDMA. Each signal at each satellite is assigned a unique pseudo random sequence, called PRN code and the carrier frequency is modulated using this sequence. Additionally navigation messages are also transmitted on this signal (see further).

Legacy GPS satellites work on two frequencies, primary L1 (1575.42 MHz) and secondary L2 (1227.6 MHz). Unencrypted 1 MHz C/A code is transmitted on L1, and encrypted military 10 MHz P(Y) code is transmitted on both L1 and L2.

One of the main advantages of multiple frequencies is the ability to calculate corrections for ionospheric delays from the received signals. To use this feature without the encryption keys for the P(Y) codes, codeless techniques have been developed that use the encrypted stream of P(Y) data and utilize timing of the bit stream. Semi-codeless techniques further exploit other properties of P(Y) code and its known relationships to C/A codes.

Modernized GPS satellites provide three additional civilian signals: L2C, L5 and a military signal M. A fourth civilian signal L1C is planned, that will be compatible with European navigation system Galileo (see Section 2.7.1). At the time of writing this text, the signals L2C and L5 are partially supported and L1C is still only planned [19].

2.2.4 Navigation Messages

Navigation messages are transmitted modulated on the ranging signals at $50\,\mathrm{bit\,s^{-1}}$. These messages contain ephemeris and clock corrections for each satellite and other data, including meta data on the ephemeris values and estimated ionospheric parameters.

See chapter 3.3 of [27] for a brief overview of navigation messages, or section 20.3.3 of [11] for the details.

PRN Sequences

As mentioned above, PRN sequences are unique to every signal and every satellite.

C/A codes have a chipping rate – the frequency of PRN code modulated on top of the carrier wave – of 1.023 MHz and P(Y) code has a chipping rate of 10.23 MHz.

Receivers typically employ a number of specialized hardware correlators that attempt to duplicate the PRN codes and match them to the received signal. The result of a successful matching is an identification of the transmitting satellite, the transmission time τ_{SV} (appearing in pseudorange definition (2.1)) modulo PRN code length and also phase and Doppler shift of the carrier wave.

2.2.5 Reference Frames

ECEF

ECEF – meaning Earth-centered Earth-fixed – is a Cartesian reference frame widely used in the GPS system. As the name suggests, the origin of ECEF reference frame is in the Earth's center of mass, the XY plane is coincident with the equatorial plane. X axis in the direction of $0°$ longitude, Y axis in the direction of latitude $90°$ East and Z axis pointing in the direction of geographical North pole. Illustration of this can be seen in Figure 2.1.

Results of ephemeris calculations are in ECEF coordinates (see Section 2.2.6) and receiver positions are calculated in ECEF as well. All later GPS calculations in this work will be in ECEF reference frame.

WGS84

The World Geodetic System 1984, defined in [20], serves as an Earth model for GPS. Exact values of the model have been updated several times in the past, but these changes have been too small to be problematic for most practical applications.

The WGS84 reference ellipsoid is used to convert between ECEF coordinates and Latitude/Longitude geodetic coordinates. The exact procedure for this conversion can be found in [20]. Height measured from the ellipsoid can be obtained during the conversion, however historically heights are measured from sea level. Sea level roughly corresponds to the geoid – a surface with constant gravity potential – also defined in WGS84.

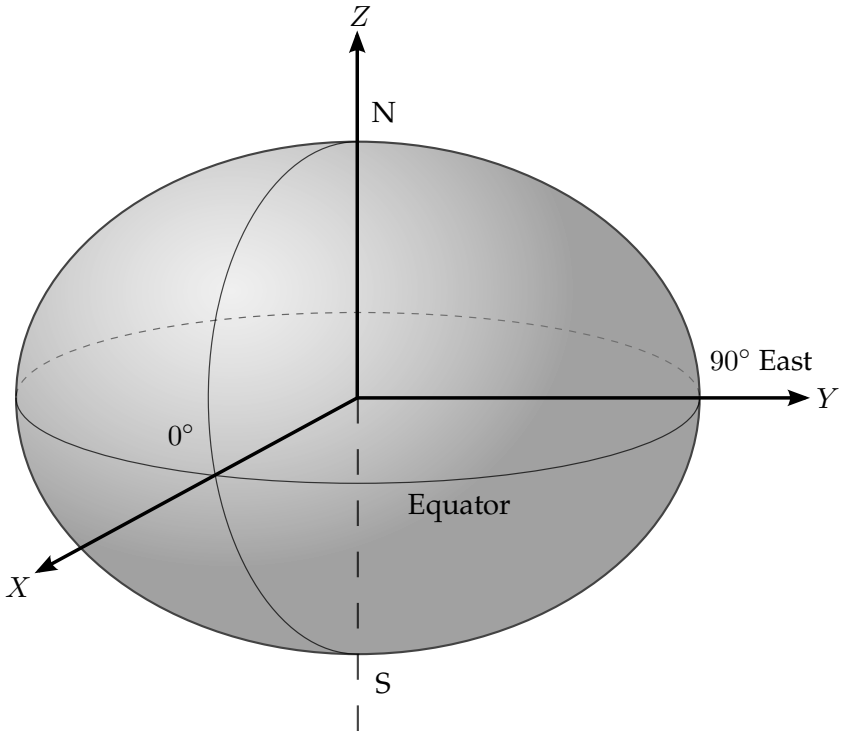


Figure 2.1: Diagram of ECEF reference frame.

2.2.6 Ephemeris

Ephemeris in the context of GPS means the data describing satellite positions and velocities at a given time. Ephemeris data are transmitted in the $50\,\mathrm{bit\,s^{-1}}$ navigation messages, together with the clock offset and drift data of the satellites, ionospheric delay estimates and other data.

Satellite orbits in the ephemeris messages are described using a set of Keplerian orbital parameters. Since in this work we will be only using ephemeris data in the ECEF reference frame, Keplerian parameter will not be discussed here. More details can be for example found in section 3.3.3 of [27], or in [15].

Precise Ephemeris

Precise ephemeris are satellite ephemeris data calculated by ground stations and distributed separately from the GPS broadcast, usually with higher precision than the data available from the GPS satellites.

One of the sources of precise ephemeris is for example [34], providing datasets with post-processed data and also with predictions for the next several hours. While the quality of these predictions is lower than of the off-line data, they are still several times more precise than the broadcast ephemerides ($5\,\mathrm{cm}$ RMS versus $100\,\mathrm{cm}$ RMS, according to [34]).

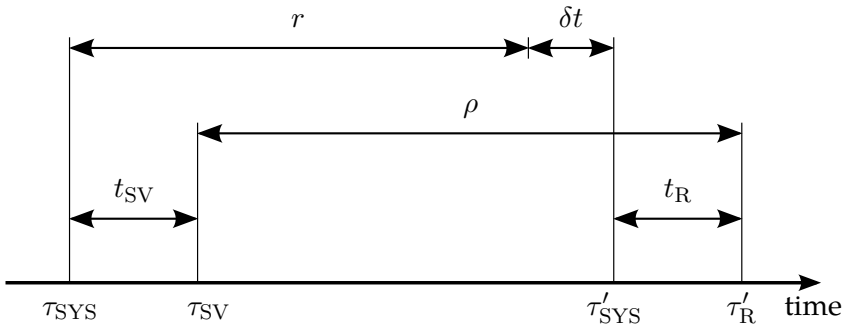


Figure 2.2: Times in pseudorange measurements.

Precise ephemeris files consist of satellite positions in ECEF coordinates and clock offsets, both sampled in 15 minutes intervals and must be interpolated before using them for navigation [30].

2.3 Obtaining Position and Velocity

2.3.1 Pseudorange

Pseudorange (ρ) is a distance that corresponds to the time taken by the ranging signal to travel from SV to the receiver, including the clock offsets:

$$\rho = \mathrm{c}(\tau'_{\mathrm{R}} - \tau_{\mathrm{SV}}) \tag{2.1}$$

Here c is the speed of light (with value $\mathrm{c} = 299\,792\,458\,\mathrm{m\,s^{-1}}$ used in GPS), τ'_{R} and τ_{SV} are the receive time referenced to the receiver internal clock and transmit time according to the satellite clock. Similarly τ_{SYS} and τ'_{SYS} are the transmit and receive times referenced to the GPS system time. Additionally, t_{SV} and t_{R} stand for the clock offset of the satellite and of the receiver, r is the real geometric distance between the satellite and receiver and δt are the signal propagation delays.

Especially in the context of pseudoranges, GPS literature often treats distance and time interchangeably, converting between them using the speed of light c. In this work, we will occasionally follow this convention as well.

In (2.1) the definition of pseudorange, the SV and receiver clocks can be converted to system time by subtracting their clock offsets:

$$\rho = \mathrm{c}(\tau'_{\mathrm{SYS}} - \tau_{\mathrm{SYS}}) + \mathrm{c}(t_{\mathrm{R}} - t_{\mathrm{SV}}) \tag{2.2}$$

It should be noted that the clock offsets change in time, and that t_{R} is receiver clock offset in the time of receiving and t_{SV} applies to the time of transmission.

Real geometric range r can be described using the transmit and receive times:

$$r + \mathrm{c}\delta t = \mathrm{c}(\tau'_{\mathrm{SYS}} - \tau_{\mathrm{SYS}}) \tag{2.3}$$

where δt are delays of the signal propagation. Together with (2.2), the previous equation gives us the basic equation for determining geometric distance from pseudorange measurements:

$$r + \mathrm{c}\delta t = \rho - \mathrm{c}(t_\mathrm{R} - t_\mathrm{SV}) \tag{2.4}$$

Figure 2.2 summarizes the relations between times in a single pseudorange measurement.

2.3.2 Position

If the position of satellites $\boldsymbol{p}_{\mathrm{SV}i} = (x_{\mathrm{SV}i}, y_{\mathrm{SV}i}, z_{\mathrm{SV}i})$ and the satellite clock offsets $t_{\mathrm{SV}i}$ at the time of transmission are known, calculating receiver position from the pseudorange measurements means solving a set of non-linear equations

$$\sqrt{(x_\mathrm{R} - x_{\mathrm{SV}i})^2 + (y_\mathrm{R} - y_{\mathrm{SV}i})^2 + (z_\mathrm{R} - z_{\mathrm{SV}i})^2} + \mathrm{c}(t_\mathrm{R} - t_{\mathrm{SV}i}) = \rho_i - \mathrm{c}\delta t_i \tag{2.5}$$

for receiver position $\boldsymbol{p}_\mathrm{R} = (x_\mathrm{R}, y_\mathrm{R}, z_\mathrm{R})$ and receiver clock offset t_R.

This corresponds to an intersection of conical surfaces in four dimensions.

Equations from (2.5) can be solved using various methods, including closed form solutions (for example [18]), iterative solutions, linearization of the equations around previous position estimates, using Kalman filters (see Section 3.2.5) or, as we will discuss in Section 4.2, Monte Carlo localization. The last two approaches also have the advantage of being able to fuse other sensor data into the solution.

Carrier Phase Tracking

Carrier phase tracking is a technique that depends on measuring the phase of carrier wave. L1 frequency has a wavelength of approximately 19 cm. Assuming the receiver can match the L1 with 1 % accuracy, then the available precision is around 2 mm compared to about 3 m for code phase (pseudorange) tracking. More detailed description of the code phase tracking errors is in Section 2.4.3.

One of the main problems with the carrier phase tracking is integer ambiguity of the carrier wave. This problem arises because, unlike the PRN signals, the carrier wave doesn't contain any code designed to help the receiver distinguish between successive periods of the signal.

Carrier phase measurements can be used to smooth pseudorange data. Details about this method can be found in section "Smooth" in [17].

2.3.3 Velocity

The simplest way of obtaining the receiver velocity is as a derivation of position, which is obtained by using any of the methods described in the preceding paragraphs. This has the advantage of requiring only a minimum of additional processing.

When the receiver position is known, its velocity can also be obtained from the Doppler shift of the received signal and known velocities of satellites. The received frequency can be approximated as

$$f = f_{\mathrm{SV}} \left(1 - \frac{v_{\mathrm{R,SV}}}{\mathrm{c}}\right) \tag{2.6}$$

where f_{SV} is the transmitted frequency and $v_{\mathrm{R,SV}}$ is the relative velocity between the satellite and the receiver. The relative velocity can be written as a dot product of an unit vector pointing from user to the satellite and velocity difference between the user and the satellite:

$$v_{\mathrm{R,SV}} = \frac{(\boldsymbol{p}_{\mathrm{SV}} - \boldsymbol{p}_{\mathrm{R}})}{\|\boldsymbol{p}_{\mathrm{SV}} - \boldsymbol{p}_{\mathrm{R}}\|} \cdot (\boldsymbol{v}_{\mathrm{SV}} - \boldsymbol{v}_{\mathrm{R}}) \tag{2.7}$$

When calculating receiver velocities, the measured values also have to be corrected for the clock drifts, both in satellite and in receiver. Clock drifts are specified in seconds per second and determine the rate of change of clock offset. Satellite clock drifts are transmitted together with their clock corrections, so we can ignore them in a similar fashion as satellite clock offsets, but receiver clock drifts must be determined together with receiver velocity.

The physically received frequency f is related to the frequency f_{R} reported by the receiver using the receiver clock drift t'_{R}:

$$f = f_{\mathrm{R}}(1 + t'_{\mathrm{R}}) \tag{2.8}$$

By putting Equations (2.6) and (2.8) together, we obtain

$$v_{\mathrm{R,SV}} = \mathrm{c}\left(1 - \frac{f_{\mathrm{R}}}{f_{\mathrm{SV}}}\right) - \mathrm{c}\frac{f_{\mathrm{R}}}{f_{\mathrm{SV}}} t'_{\mathrm{R}} \tag{2.9}$$

This equation arises for every satellite in view and as with pseudoranges and positions, there are several ways to obtain the receiver velocity from them. In this work we will only use the Doppler measurements as an input to Monte Carlo localization and we will not discuss the other methods.

2.4 Measurement Errors

Previous text assumed that all measurements in the GPS system can be made accurately, but practically all the segments of the GPS system introduce errors.

2.4.1 User Equivalent Range Error

User equivalent range error (UERE) characterizes the effective accuracy of a pseudorange measurement. UERE is defined as a sum of errors caused by different parts of the GPS system.

UERE and its components are usually assumed to be zero mean Gaussian variables, mutually independent both between the error components of a single measurement and between satellites.

2.4.2 Dilution of Precision

Dilution of precision (DOP) is a value that describes the effects of the satellite geometry on the precision of the calculated fix. There are several types of dilution of precision, specifying errors in different directions. These are PDOP (position in all directions), HDOP (horizontal error), VDOP (vertical error) or TDOP (time error).

In theory, standard deviation of the completed fix should have a linear dependency on UERE and DOP (see equation (2.13)), but the actual position error is typically slightly lower, since the ionospheric errors are correlated.

In Section 4.1 we are discussing dependence of the position error on HDOP value and Figure 4.1 shows the experimental data.

Definition

The formal definition of DOP values is based on linearization of pseudorange equations in Section 2.3.2:

$$\boldsymbol{H}\boldsymbol{\Delta x} = \boldsymbol{\Delta \rho} \tag{2.10}$$

Here $\boldsymbol{H}$ is a matrix of unit vectors pointing from the linearization point to the satellites, $\boldsymbol{\Delta \rho}$ is a vector containing the difference between pseudoranges in linearization point and the real measured pseudoranges and finally $\boldsymbol{\Delta x}$ is a vector describing the offset of receiver position from the linearization point. To simplify the final definitions of DOP values, all calculations here are performed in a local reference frame with the z axis pointing up from the linearization point.

Equation (2.10) is solved for $\boldsymbol{\Delta x}$ using least squares and as a next step, pseudorange and position errors are taken into account. This yields an expression relating position error to pseudorange error:

$$\boldsymbol{\delta x} = (\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{\delta \rho} \tag{2.11}$$

Where $\boldsymbol{\delta x}$ and $\boldsymbol{\delta \rho}$ are position errors and pseudorange errors.

As a next step, pseudorange and position errors are taken into account and after some modifications, relations between position and pseudorange error are expressed. All pseudorange errors are assumed to be independent, Gaussian and identically distributed.

The covariance matrix of $\boldsymbol{\delta x}$ can then be expressed as

$$\operatorname{cov}(\boldsymbol{\delta x}) = (\boldsymbol{H}^T\boldsymbol{H})^{-1}\sigma^2_{\mathrm{UERE}} = \begin{pmatrix} \sigma_x^2 & \sigma_{xy}^2 & \sigma_{xz}^2 & \sigma_{xt}^2 \\ \sigma_{xy}^2 & \sigma_y^2 & \sigma_{yz}^2 & \sigma_{yt}^2 \\ \sigma_{xz}^2 & \sigma_{yz}^2 & \sigma_z^2 & \sigma_{zt}^2 \\ \sigma_{xt}^2 & \sigma_{yt}^2 & \sigma_{zt}^2 & \sigma_t^2 \end{pmatrix} \tag{2.12}$$

The covariance matrix only depends on satellite geometry as described by $\boldsymbol{H}$ and UERE.

DOP values are defined as follows:

$$\sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2} = \mathrm{PDOP}\sigma_{\mathrm{UERE}} \tag{2.13a}$$

$$\sqrt{\sigma_x^2 + \sigma_y^2} = \mathrm{HDOP}\sigma_{\mathrm{UERE}} \tag{2.13b}$$

$$\sqrt{\sigma_z^2} = \mathrm{VDOP}\sigma_{\mathrm{UERE}} \tag{2.13c}$$

$$\sqrt{\sigma_t^2}/\mathrm{c} = \mathrm{TDOP}\sigma_{\mathrm{UERE}} \tag{2.13d}$$

DRMS

Distance root mean square is an accuracy metric, somewhat similar to standard deviation of a distribution. It is a single value describing the accuracy of a 2D position fix.

DRMS is defined as

$$\mathrm{DRMS} = \sqrt{E(\|\boldsymbol{\delta R}\|^2)} \tag{2.14}$$

where $\boldsymbol{\delta R}$ is a horizontal component of the position error.

Theoretically DRMS can be obtained from HDOP values:

$$\mathrm{DRMS} = \mathrm{HDOP}\sigma_{\mathrm{UERE}} \tag{2.15}$$

[40] offers an alternative expression for obtaining DRMS from HDOP:

$$\mathrm{DRMS} = \sqrt{(a\mathrm{HDOP})^2 + b^2)} \tag{2.16}$$

where a and b are parameters that are fitted to measured data. In section 4.1.2 we show how this expression fits our experimental data.

$2 \times \mathrm{DRMS}$ is often used as an approximation for a radius containing $95\,\%$ of measurements.

2.4.3 Error Sources

The following text describes the major sources of measurement errors and also a way of removing or at least limiting them. Apart from errors mentioned in this section, there are many more sources, but they cause inaccuracies that are negligible for single receiver applications discussed in this text. An overview can be found in [16].

Geometric Distribution of Satellites

Geometric distribution of the satellites significantly influences final precision of the position. If satellites are approximately uniformly distributed around the receiver, the localization error is smaller than when satellites are concentrated in a single direction. This factor of GPS localization precision is characterized by the parameter HDOP mentioned earlier (see Section 2.4.2).

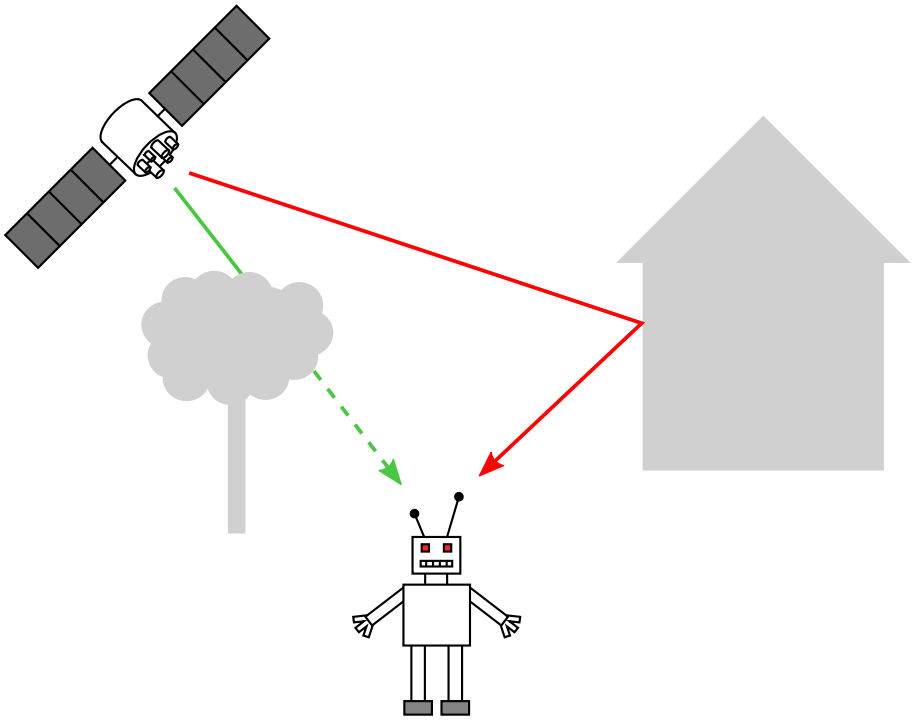


Figure 2.3: Multipath and shadowing.

Match Accuracy and Receiver Delays

When the receiver matches the received signal to expected PRN codes, an important question arises – how accurate the matching is. Often cited value is 1 %, which for 1.023 MHz chipping rate gives about 2.9 m. This kind of errors can be improved by using higher quality receiver hardware or by smoothing with carrier phase measurements (see section 2.3.2).

Delays also occur in receiver signal processing, both in the electronics and in software, but if these delays are constant for all pseudorange measurements, they will be removed together with receiver clock delay.

Multipath

Multipath errors appear when the signal from the satellite reaches the receiver through multiple paths of different length. They are typically caused by buildings surrounding the receiver position, reflective surfaces surrounding the antenna (e.g. wings in an aircraft mounted GPS) or Earth's surface when including satellites with low elevation angle in the solution.

An illustration of this effect and of shadowing is shown in Figure 2.3. The signal drawn in green follows a direct path from the satellite and is shadowed. The red multipath signal is reflected from a building, making its traveled path longer.

If the signal following the direct path is received, GPS receivers can easily detect and remove large delays caused by multipath. A more problematic case arises when the multipath delay is relatively small, because this distorts receiver's attempts to correlate the received signal with the internal reference signal. It is possible, however, to use even non-line of sight measurements, as described in [3].

Section 6.3 of [15] contains a detailed overview of multipath effects.

Shadowing

Shadowing is another phenomenon related to multipath, caused by the signal passing through obstacles, like buildings or foliage. Shadowing combined with multipath affects the relative received power of direct and multipath signals, in extreme cases even causing the multipath to be the only received signal from a given satellite.

Multipath errors can be mitigated with hardware modifications, like antenna designs that suppress signals from suspicious angles, better antenna placement or coating nearby reflective surfaces with RF absorptive materials. Another option is to employ better signal processing in receivers.

When only complete measurements are available (as is the case in the implementation part of this text), satellites may be dropped from the solution if the elevation angle is too low to avoid this kind of errors. A more complex method of dealing with multipath errors is described in [38], where the error model of a satellite measurement is modified if it is suspected to be corrupted by multipath error.

Tropospheric Delays

Tropospheric delays describe signal delays in the non-ionized layers of Earth's atmosphere.

Troposphere is the non-ionized layer of atmosphere closest to the surface of the Earth. The stratosphere – another non-ionized layer of atmosphere – causes the same type of delays and is usually included in the therm tropospheric delay.

Tropospheric delays are modeled as consisting of a wet and a dry component, and when left uncompensated, they amount to between $2.4\,\mathrm{m}$ and $25\,\mathrm{m}$ depending on satellite elevation angle, the dry component responsible for about $90\,\%$ of the delay [15].

There are several models for estimating tropospheric delays depending on satellite elevation angle, altitude of the receiver, atmospheric pressure, temperature and water vapor pressure.

In the implementation part of this work we use the model outlined in [23] with hard-coded meteorologic parameters.

Ionospheric Delays

Compared to tropospheric delays, signal delays caused in the ionosphere are larger and harder to predict.

An interesting property of ionospheric delays is, that they are correlated, both in measurements from different satellite to a single receiver and in measurements from a single satellite to multiple receivers. The later property is utilized in differential GPS (Section 2.5.2).

Since the ionospheric delay is a function of signal frequency and total electron count in atmosphere [15], it can be almost completely eliminated when using multi frequency receivers. Single frequency receivers, are however forced to use less accurate models of the ionosphere and ionospheric parameters (Klobuchar model for GPS [21] or NeQuick model for Galileo [22]). Ionospheric correction estimates are also transmitted in the GPS navigation messages [11].

Ephemeris and SV Clock Errors

Ephemeris and satellite clock corrections are periodically uploaded by the control segment, however the residual errors increase with the age of upload. According to [15], 1σ pseudorange error is typically about $0.8\,\mathrm{m}$ and up to $4\,\mathrm{m}$ for clock errors.

As discussed in Section 2.2.6, this class of errors can be avoided or at least decreased by using precise ephemeris at the expense of requiring internet connection during operation.

Relativistic Effects

The most famous effect of relativity is the frequency shift of the satellite clock when received on the ground. This frequency shift happens because of the satellites velocity relative to the user and also because satellite orbits are further from Earth's mass and therefore less affected by the space time curvature caused by it.

To compensate, satellite clock frequency is set to $10.229\,999\,995\,43\,\mathrm{MHz}$ before launch, so the observed frequency is $10.23\,\mathrm{MHz}$ [11].

2.5 Methods for Improving GPS Precision

2.5.1 Kalman Filters

Kalman filters are often used in the GPS receivers to merge past data with incoming measurements or to combine the GPS position and velocity estimation with another source of measurements, for example inertial data.

It works by representing a current estimate of the receiver position together with its covariance matrix and updating it based on the incoming measurements. For more detailed description of how Kalman filters operate see Section 3.2.5.

2.5.2 Differential GPS

Differential GPS is a method to improve GPS accuracy using at least one reference station. DGPS exploits the correlations of GPS errors and is capable of removing or decreasing satellite clock errors, ephemeris errors, tropospheric and ionospheric errors.

DGPS systems may operate with baseline distances from hundreds of meters to several thousand kilometers. They may be used to calculate precise absolute positions or positions relative to the reference station. Accuracy of DGPS may range from several decimeters for code based systems to several millimeters for carrier based systems.

Reference stations must feed their data to the receivers, which correct their own measurements base on them. This need of infrastructure and reliable wireless connection makes DGPS impractical for many uses.

Differential GPS is often used in surveying (off-line, high precision) and this use is discussed in [27].

2.5.3 Assisted GPS

Assisted GPS refers to the techniques used typically on PDAs and cell phones, that improve operation of GPS by offloading work from the receiver chip to a remote assistance server.

The assistance server has a good satellite reception and more computing power than the device and can supply the client system with complete position fixes based on uploaded GPS signal sample, ephemeris data which can then be used to speed up cold start of the client device or ionospheric parameters, forming simple DGPS system and increasing the fix precision.

2.6 Protocols

Most of the time a consumer GPS receiver is used as a black box that provides position estimates when power and possibly external antenna is connected. In more complicated use cases, such as measurement domain integration of GPS signals with other sensors discussed in Section 4.2 of this work, lower level data from the location estimation process may be used, but only as provided by software in the GPS receiver. Many communication protocols are in use, which providing position, velocity, clock data and other information. Several of the protocols are mentioned in the following paragraphs.

2.6.1 NMEA 0183

NMEA 0183 is a de-facto standard protocol for consumer grade GPS receivers. It is a text-based proprietary format developed by National Marine Electronics Organization, but has been reverse engineered and the specification is known and widely used [5].

The protocol by itself is designed for communication of marine electronic equipment and isn't specific to GPS receivers. It consists of a large number of sentences, some of which are intended to be used for position, velocity and time solutions of GPS navigation.

The problem with the NMEA protocol is, that a different subset is implemented in almost every receiver and there is no access to lower levels of the GPS signal processing.

2.6.2 RINEX

The Receiver Independent Exchange Format [13] is an exchange format for GPS and other satellite navigation systems.

RINEX is typically used for post processing as it can contain various data that are not known during navigation, for example detailed ionospheric models or high precision ephemeris.

2.6.3 Proprietary Protocols

Many GPS receivers extend the NMEA protocol using proprietary sentences or implement custom protocols. An example of this is the SiRF binary protocol [32], which is also used in the implementation part of this work.

The use of proprietary protocol is often the only way to get the complete functionality from the receiver.

2.7 Similar Systems

GPS is not the only GNSS currently operational. Other countries than USA have also created navigation systems, and several regional satellite navigation systems also exist. Section 1.5 of [24] contains a more detailed overview.

GLONASS [7] is a Russian navigation system, in principle similar to GPS. It is currently undergoing a modernization and is planned to be compatible with GPS and Galileo.

Another example of a current GNSS system is the Chinese CNSS / COMPASS [31], also known as BeiDou-II. The system is planned to be operational by 2020.

2.7.1 Galileo

Galileo [1] is a project by the European Union to create a global navigation system, which is probably of the most interest to current GPS users.

It is designed to be independent of GPS and to offer multiple levels of access worldwide including an open access navigation, however it will be compatible with GPS receivers that support the modernized L1C signals. The Galileo constellation will consist of 30 satellites, and will provide several types of services, from free public service to safety-critical services.

Technically, Galileo is fairly similar to GPS. Their compatibility is provided by using the same geodetic and time reference frames and as mentioned above, by using a signal compatible with the GPS L1C. This means, that a GPS receiver supporting this signal will be able to seamlessly use both systems combined.

Galileo, however, cannot be used practically yet, which is why we concentrate on GPS. Moreover, measurements from Galileo satellites will suffer from the same errors as their GPS counterparts, which makes this system interchangeable with the GPS for the purposes of this work.

3 Monte Carlo Localization

The following chapter defines the localization problem, provides an overview of Markov localization and Kalman filters and finally describes the Monte Carlo localization algorithm.

3.1 Localization

Localization of a mobile robot is a task of estimating position (and possibly also other state) of the robot based on performed actions and sensor readings. To navigate its environment reliably, an autonomous mobile robot needs a relatively precise estimate of its position.

Localization methods can be categorized according to many different criteria. One of the possibilities is the categorization based on whether the environment is static or dynamic.

Another dimension along which the localization algorithms can be grouped is the ability of the localization algorithm to modify behavior of the robot in order to find the current position faster or with more precision.

Next, localization can mean either position tracking, where the robot knows its initial position and the task is only to update the estimate and handle relatively small errors, or global localization, where the robot has to find its position from scratch and recover from serious localization errors.

In this work we will focus only on passive localization in static environment. Since Monte Carlo localization is a topic of this text we will be favoring global localization, although typically GPS is used with Kalman filters which are only capable of tracking the estimate locally.

3.2 Markov Localization

Markov localization [9, 6] is a recursive probabilistic algorithm that estimates state of the robot based on its actions and possibly noisy measured data.

3.2.1 Assumptions

Markov localization assumes that the state of the robot only depends on the state and action performed in the previous time frame:

$$\Pr(X_i = x \mid X_{1,\dots,i-1}, a_{1,\dots,i-1}, o_{1,\dots,i-1}) = \Pr(X_i = x \mid X_{i-1}, a_{i-1}) \tag{3.1}$$

and that each observation only depends on the current state:

$$\Pr(o_i \mid X_{1,\dots,i}, a_{1,\dots,i-1}, o_{1,\dots,i-1}) = \Pr(o_i \mid X_i) \tag{3.2}$$

3.2.2 Derivation

The algorithm computes the probability density of the robot state, called belief.

$$\mathrm{Bel}(X_i = x) = \Pr(X_i = x \mid o_{1,\dots,i}, a_{1,\dots,i-1}) \tag{3.3}$$

Here X_i is the random variable representing the robot's state, o_i is the observed sensor input in time step i and a_i is the action the robot performs in the time step i, after it measures o_i.

Using Bayes rule, Equation (3.3) can be transformed to

$$\mathrm{Bel}(X_i = x) = \frac{\Pr(o_i \mid X_i = x, o_{1,\dots,i}, a_{1,\dots,i-1}) \Pr(X_i = x \mid o_{1,\dots,i}, a_{1,\dots,i-1})}{\Pr(o_i \mid o_{1,\dots,i-1}, a_{1,\dots,i-1})} \tag{3.4}$$

Note that the denominator $\Pr(o_i \mid o_{1,\dots,i-1}, a_{1,\dots,i-1})$ only serves as a normalization constant. In further equations it will be replaced by η^{-1}. Other than that, we can use the independence assumptions made at the beginning of this section and simplify (3.4) to

$$\mathrm{Bel}(X_i = x) = \eta \Pr(o_i \mid X_i = x) \Pr(X_i = x \mid o_{1,\dots,i}, a_{1,\dots,i-1}) \tag{3.5}$$

By integrating over all possible states in time $i-1$, the rightmost term in Equation (3.4) can be expanded in a following way:

$$\begin{aligned}&\Pr(X_i = x \mid o_{1,\dots,i}, a_{1,\dots,i-1}) = \\ &\int \Pr(X_i = x \mid X_{i-1} = x', o_{1,\dots,i}, a_{1,\dots,i-1}) \Pr(X_{i-1} = x' \mid o_{1,\dots,i-1}, a_{1,\dots,i-1}) \,\mathrm{d}x'\end{aligned} \tag{3.6}$$

After substituting the definition of belief from Equation (3.3) and another use of the independence assumptions we get

$$\Pr(X_i = x \mid o_{1,\dots,i}, a_{1,\dots,i-1}) = \int \Pr(X_i = x \mid X_{i-1} = x', a_{i-1}) \,\mathrm{Bel}(X_{i-1} = x') \,\mathrm{d}x' \tag{3.7}$$

Finally, substituting into (3.5) gives us the recursive equation

$$\mathrm{Bel}(X_i = x) = \eta \Pr(o_i \mid X_i = x) \int \Pr(X_i = x' \mid X_{i-1} = x', a_{i-1}) \,\mathrm{Bel}(X_{i-1} = x') \,\mathrm{d}x' \tag{3.8}$$

We will call the conditional density $\Pr(o_i \mid X_i = x)$ *sensor model* and the density $\Pr(X_i = x \mid X_{i-1} = x', a_{i-1})$ *motion model*. The sensor model describes the probability of observing o_i at given time. It implicitly contains a map of the environment and models interactions of sensors with the environment. The action model contains information about how the robot's actions relate to changes in its state.

3.2.3 Algorithm

The algorithm for Markov localization operates in two alternating phases: prediction phase, in witch the algorithm incorporates a performed action into the belief (and therefore predicts the state after the action) and correction phase which updates the measurements based on the observed sensor measurements.

Prediction

The prediction phase uses the action model to obtain predictive density based on the action performed and the previous belief, by integrating over all possible states at time $i - 1$. The following equation is an exact copy of (3.7), included here for completeness.

$$\Pr(X_i = x \mid o_{1,\dots,i}, a_{1,\dots,i-1}) = \int \Pr(X_i = x \mid X_{i-1} = x', a_{i-1}) \operatorname{Bel}(X_{i-1} = x') \, \mathrm{d}x' \tag{3.9}$$

Correction

In the correction phase, the belief is regenerated based on the predictive density and the sensor model.

$$\operatorname{Bel}(X_i = x) = \eta \Pr(o_i \mid X_i = x) \Pr(X_i = x \mid o_{1,\dots,i}, a_{1,\dots,i-1}) \tag{3.10}$$

3.2.4 Density Representations

The algorithm as described above doesn't tell us how to represent the probability densities encountered. Several specializations of Markov localization exist, differing mainly in how the current state estimation is represented. Examples include Gaussian distribution in Kalman filters, grid based algorithms [9], or representing the distribution using samples in Monte Carlo localization (see Section 3.3).

3.2.5 Kalman Filter

Kalman filter [14, 39] is a popular state estimator, often used in robotics for localization. It can be viewed as a closed form solution of Markov localization where both the action model and the measurement model are linear and Gaussian [6].

$$\Pr(X_i = x \mid X_{i-1} = x', a_{i-1}) \sim \mathcal{N}(A_{i-1}x' + B_{i-1}a_{i-1}, Q_{i-1}) \tag{3.11}$$

$$\Pr(o_i \mid X_i = x) \sim \mathcal{N}(H_i x, R_i) \tag{3.12}$$

In Equations (3.11) and (3.12) the matrix A_i describes the change of state if there was no control input and no noise, matrix B_i contains the influence of action a_i and H_i relates state to measured values. Q_i and R_i are covariance matrices of the action model and the measurement model.

Because the initial belief, the action model and the measurement model are all Gaussian, belief will always remain Gaussian as well. This means that only the mean value and covariance matrix need to be stored, making Kalman filter very efficient, requiring only several matrix operations in each step.

Non linear motion and measurement models can be approximated using a first order Taylor expansion to form an extended Kalman filter (EKF, [39]).

As an illustration of operations of a Kalman filter, the update rules are included. For more detailed description see [39].

Prediction

$$\bar{x}_i^- = A_{i-1}\bar{x}_{i-1} + B_{i-1}a_{i-1} \tag{3.13}$$

$$P_i^- = A_{i-1}P_{i-1}A_{i-1}^{\mathrm{T}} + Q_{i-1} \tag{3.14}$$

Here $\bar{x}_i$ and P_i are mean and covariance describing $\mathrm{Bel}(X_i)$. $\bar{x}_i^-$ and P_i^- are mean and covariance of the predictive density $\Pr(X_i \mid o_{1,\dots,i}, a_{1,\dots,i-1})$

Correction

$$K_i = P_i^- H_i^{\mathrm{T}}(H_i P_i^- H_i^{\mathrm{T}} + R_i)^{-1} \tag{3.15}$$

$$\bar{x}_i = \bar{x}_i^- + K_i(o_i - H_i\bar{x}_i^-) \tag{3.16}$$

$$P_i = (I - K_i H_i)P_i^- \tag{3.17}$$

3.3 Monte Carlo Localization

Monte Carlo localization (MCL, [4]) is a Monte Carlo method for robot localization. It is a version of Markov localization, that approximates the belief using a set of samples drawn from it.

Advantages of MCL include a possibility to represent arbitrary shapes of probability densities as opposed to Gaussian distribution in the widespread Kalman filters and the related ability to localize the robot globally. Compared to the grid based methods MCL

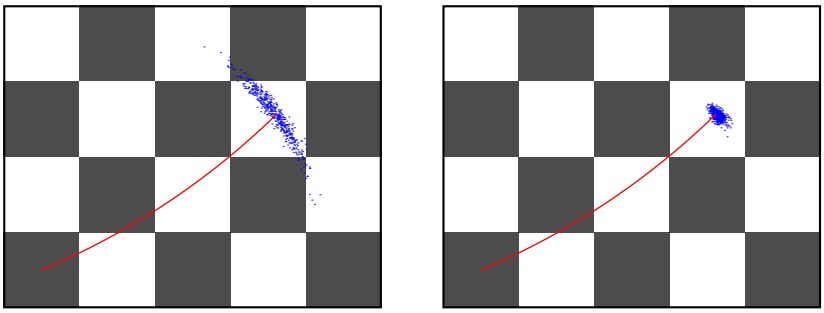


Figure 3.1: Illustration of MCL sample cloud.

automatically focuses most of the computing power to the highly probable regions. This algorithm is also fairly easy to implement and the action and sensor models in sampling and probability forms are simpler than covariance matrices in Kalman filters.

On the other hand, Monte Carlo localization is more CPU intensive than Kalman filters and the particle approximation doesn't work well with too precise sensors, because if the observation model PDF becomes too "sharp" peaks, it has a high probability of missing all samples during the correction phase and effectively ignoring the measurement.

Informally the basic MCL can be described as follows: Keep n samples of hypothetical robot states. In the prediction phase move each sample according to the action performed with the noise corresponding to uncertainty on the action's result (e.g. wheel slip) added. Correction calculates how probable the incoming measurement is for every sampled state and set this probability as the sample's weight. Finally in the resampling phase throw away samples with low weights, and add copies of the highly weighted samples instead.

Figure 3.1 shows example of two identical simulated robots on a chessboard. Red lines show paths traveled by the robots and blue dots represent MCL samples. Both of the robots use odometry data to estimate position and the robot on the right senses color of the tile underneath it as well.

3.3.1 Algorithm

MCL keeps a set of samples $S_i = \{s_i^k \mid k = 1, \ldots, n\}$ drawn from the distribution $\Pr(X_i = x \mid X_{i-1}, o_i, a_{i-1})$ to represent the belief and performs prediction, correction and resampling phases for every time frame.

Pseudocode in Figure 3.2 contains the basic structure of Monte Carlo localization. The algorithm reads inputs from the environment by calling functions `getAction()` and `getObservation()`, that return the last performed action and the last measured observation. Output is made available to the rest of the robot's software in function `outputPosition()`.

```
samples ← initialize()
repeat
    samples ← predict(samples, getAction())
    weights ← correct(samples, getObservation())
    samples ← resample(samples, weights)
    outputPosition(samples)
end

function initialize :
    Input: nothing
    Output: list of samples
    samples ← emptyList
    for i ← 1 to n do
        append(samples, randomSample())
    end
    return samples
end

function predict :
    Input: list of samples, performed action
    Output: modified list of samples
    foreach s ∈ samples do
        s ← sampleFromActionModel(s, action)
    end
    return samples
end

function correct :
    Input: list of samples, observation
    Output: list of weights
    weights ← emptyList
    foreach s ∈ samples do
        append(weights, observationProbability(s, observation))
    end
    return weights
end
```

Figure 3.2: Monte Carlo localization algorithm

Prediction and correction steps in the algorithm utilize the motion model through `sampleFromActionModel()` and sensor model using `observationProbability()`. The function `sampleFromActionModel()` returns a random predicted sample based on a previous sample and an action, `observationProbability()` gives the probability of an observation in a given state.

The `initialize()` function is an example of how MCL can be used to localize a robot without knowledge of its previous location by starting with samples distributed uniformly through the environment.

In a practical implementation it is often useful to "invert" the algorithm and have external code pass events with performed action and sensor readings.

Prediction

- Create a new sample set $S'_i = \{s'^k_i\}$ by drawing a new sample s'^k_i from the action model $\Pr(X_i = x \mid X_{i-1} = s^k_{i-1}, a_{i-1})$ for every sample in S_{i-1}.

This is stratified sampling from the empirical predictive density

$$\hat{\Pr}(X_i = x \mid X_{i-1}, a_{i-1}) = \sum_{k=1}^{n} \Pr(X_i = x \mid X_{i-1} = s^k_{i-1}, a_{i-1}) \tag{3.18}$$

which is used instead of the predictive density from (3.9).

Correction

- Using the sensor model, calculate weight $w^k_i = \Pr(o_i \mid X_i = s'^k_i)$ for each sample s'^k_i in the predictive sample set S'_i.

Ideally in the correction phase the algorithm would draw samples directly from $\Pr(o_i \mid X_i = x) \Pr(X_i = x \mid X_{i-1}, a_{i-1})$, however there is no easy way to achieve this. Instead, Monte Carlo localization uses sampling / importance resampling (SIR, [33]) to approximate sampling from $\Pr(o_i \mid X_i = x)\hat{\Pr}(X_i = x \mid X_{i-1}, a_{i-1})$ using the set S'_i. This is achieved by weighting the samples as described earlier and resampling according to these weights in the next phase.

The weights are obtained by dividing the target density by the available density:

$$w^k_i = \frac{\Pr(o_i \mid X_i = s'^k_i)\hat{\Pr}(X_i = s'^k_i \mid X_{i-1}, a_{i-1})}{\hat{\Pr}(X_i = s'^k_i \mid X_{i-1}, a_{i-1})} = \Pr(o_i \mid X_i = s'^k_i) \tag{3.19}$$

Resampling

- Draw n new samples s^k_{i+1} from S'_i by placing weight $\{w^k_i\}$ on s'^k_i.

The resampling phase is necessary to finalize the SIR procedure from the correction phase. It can be performed in $\mathrm{O}(n)$, using the systematic resampling algorithm described in [2].

3.3.2 Modifications

Adaptive Sample Size

During global localization or tracking failure MCL needs a relatively large number of samples to localize correctly. The required number of samples, however, decreases during position keeping.

An improvement of the MCL algorithm is to vary the sample size dynamically. This can be done by using likehood sampling [10], that works by resampling not until a fixed number of samples is generated, but until the sum of unnormalized weights exceeds a given threshold. Another, more involved approach, using Kullback-Leibler distance in KLD-sampling algorithm is described in [8].

Delayed Resampling

It is possible to keep sample weights during the whole operation of the algorithm. This means that the resampling phase can be performed only once for several prediction and correction steps for example to save computing power. Various criteria for starting resampling may be chosen, examples include a fixed number of prediction / correction steps, or when the effective number of particles drops under a predefined threshold. This is explained in detail in [2].

Mixture MCL

Mixture MCL [35] is a modification of Monte Carlo localization that solves the problem with too precise observations. The modified algorithm uses a dual algorithm to MCL – sampling from the observation model and correction from the motion model – and combines it with regular Monte Carlo localization.

4 GPS Used in Monte Carlo Localization

In this chapter we will show two approaches to using GPS receivers with MCL. First we will concentrate on the approach based on high level data that are available from the standard WGS84 protocol. Next we will show an original algorithm that utilizes the individual pseudorange measurements each as an independent correction input.

For an autonomous outdoor robot a GPS seems to be a good candidate for a source of absolute position information. Positions reported by low cost GPS receivers, however, suffer from large errors, which make it impractical to be used as a main localization sensor. Monte Carlo localization is an algorithm that has the ability to deal with noisy inputs and is relatively simple to both understand and implement. Moreover only a relatively simple model of the input sensor is necessary to work with MCL, enabling it to calculate the GPS position solutions without requiring a lot of theory otherwise necessary to use GPS at the lower level.

This chapter concentrates on tasks common to all GPS receivers. Problems and implementation decisions encountered with the SiRF receiver during the implementation part are addressed in Chapter 5. Specific values in this chapter are calculated from experimentally measured data (see Appendix B).

GPS augmentation systems (like EGNOS or WAAS) were not used for tests in this work to keep the sensor models simple. Extending the algorithms provided here to utilize GPS augmentation is an opportunity for further work, although it should not introduce dramatic changes to the framework.

4.1 Position Domain Integration

Integrating the measurements in position domain means processing the complete fixes and treating the GPS receiver as a source of absolute position and velocity information. This processing can be performed in ECEF reference frames, or, as is the case in this work, in 2D plane.

This approach has the advantage of being very simple, because most of the work is done in the dedicated hardware of the GPS receiver. On the other hand information is

discarded when the fix is converted to the simple latitude / longitude / HDOP format, which may decrease the accuracy of the localization. Furthermore consumer-grade GPS receivers often employ filters tuned to specific properties of the car, boat, plane or other platform intended to carry the unit. These may for example ignore speeds below a certain threshold or limit allowed accelerations.

It would also be possible for position domain integration to utilize the speed information from the GPS (NMEA message GPRMC contains velocity in knots and track angle [5]). For this work, however, we choose not to employ it, to avoid the complexity of converting track course and speed to reference frames usable for robot navigation.

4.1.1 Robot State

For position domain integration we work in a two dimensional coordinate system on a surface of Earth, with X axis heading eastward and Y axis heading northward. WGS84 inputs are transformed to this reference frame using orthogonal projection and then this projected position and HDOP are used to modify weights of the samples. Details of this can be found in Section 5.3.1.

No other variables are required in the robot's state for this method. This makes it possible to seamlessly integrate it into existing localization framework. Adding more state, however might improve the accuracy, one such option is discussed in Section 4.1.3.

4.1.2 Sensor Model

Since the position domain integration as used here has no persistent variables in the robot state, it does not need any operations during the prediction step of MCL.

Our sensor model for the correction phase closely follows [40]. We are operating only in two dimensions and the horizontal component of the position error $\boldsymbol{\delta R}$ is modeled using Rayleigh distribution parameterized with HDOP of the measurement:

$$\Pr(\|\boldsymbol{\delta R}\| < x \mid \mathrm{HDOP}) = 1 - e^{-x^2/2\sigma(\mathrm{HDOP})^2} \tag{4.1}$$

Parameter $\sigma(\mathrm{HDOP})$ of the distribution is chosen to fit the DRMS values using a maximum likehood estimate

$$\sigma(\mathrm{HDOP}) = \frac{\mathrm{DRMS}(\mathrm{HDOP})}{\sqrt{2}} \tag{4.2}$$

The experimentally measured data were projected to a plane and distance errors were calculated as a difference of the point position from a mean position. These errors were then fitted to the theoretical linear model and to the non-linear model from [40].

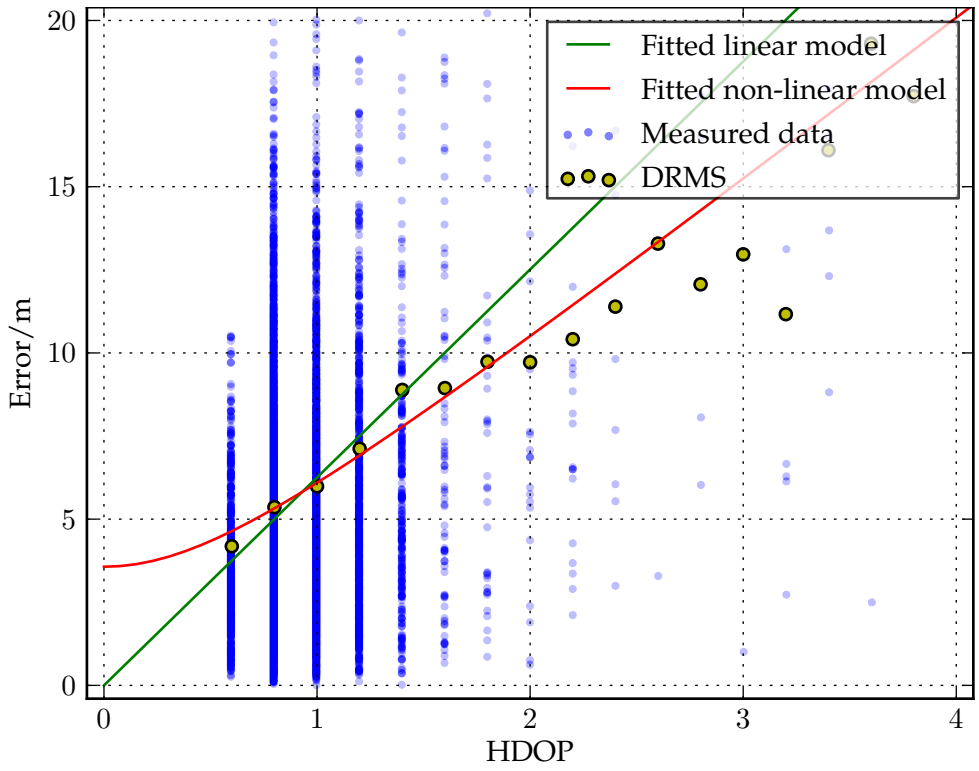


Figure 4.1: Measurement errors vs. HDOP.

Both were fitted to the data using least squares weighted with counts of samples for each HDOP, resulting in the expressions

$$\mathrm{DRMS(HDOP)/m} = 6.255\mathrm{HDOP} \tag{4.3}$$

and

$$\mathrm{DRMS(HDOP)/m} = \sqrt{(4.941\mathrm{HDOP})^2 + 3.568^2} \tag{4.4}$$

Figure 4.1 is a visualization of fitting of these two models, blue dots representing the measured data, yellow dots the DRMS values. The green line shows the theoretical linear model and the red curve is the non linear model. Limited resolution of the HDOP values in input data are visible in the plot. This figure also shows the precision available when using simple NMEA data – DRMS error of roughly 7 m. Another interesting fact is, that only relatively low HDOP values were encountered. Figure B.1 shows this in a more pronounced way.

To obtain the final sensor model, we must express the PDF of Rayleigh distribution:

$$f(x) = \frac{x}{\sigma^2} e^{-x^2/2\sigma^2} \tag{4.5}$$

This probability distribution is used as the sensor model in Section 3.3.

Figure 4.2 shows a pseudo code implementation of a function that provides a sensor model to MCL algorithm from Chapter 3 based on the fitted non-linear model.

```
function observationProbabilityGPS :
    Input: sample, observation
    Output: probability

    projected ← project(observation.latLon)
    HDOP ← observation.HDOP

    positionError ← abs(projected − sample)
    DRMS² ← (4.941HDOP)² + 3.568²

    return 2 * positionError / DRMS² * exp(-positionError²/DRMS²)
end
```

Figure 4.2: Correction algorithm for position domain integration of GPS measurements

4.1.3 Correlation of Position Errors

A problem that is hard to avoid with this approach is that position errors of the receiver output are correlated. This is in part because errors of the individual satellite measurements are correlated, another major reason is the "inertia" added by the Kalman filter in receiver firmware. This kind of correlation obviously breaks the independence assumptions of Markov localization defined in Section 3.2.1.

An attempt at mitigating this could be done by estimating the error as part of the robot's state, possibly removing atmospheric effects and some of the low pass filtering properties of the Kalman filter in the receiver. These options, however, will not be explored in this work.

4.2 Measurement Domain Integration

Measurement domain integration takes each pseudorange measurement as an input and combines it with other sensor data.

In Monte Carlo localization, the GPS data are used in a similar way as for example ultrasound ranging beacons would be used. As a first step measured pseudorange is corrected for measurement errors. Sample weights are then modified based on the corrected distance compared to expected distance. The use of pseudoranges forces the additional complexity of estimating the receiver clock offset, which is needed in order to transform pseudoranges to measured geometric ranges.

The main reason for attempting to use the GPS in this way is to add more information into the localization process and to have more control over assumptions made during processing of the GPS data.

To improve the precision of the localization, we also estimate residual errors of each satellite, covering all of the slowly changing errors that are present in each satellite's transmission and are not explicitly corrected elsewhere in the process. This includes

satellite clock inaccuracies, unmodeled remains of ionospheric and tropospheric errors and to some extent also satellite ephemeris errors. While this is not strictly necessary and the localization can work even when only estimating receiver clock offset and drift, this significantly improves precision. In the experimental data estimation of residual errors decreased one sigma pseudorange error from 18.586 m to 7.062 m.

As with the Position domain approach, the model was derived from measurements in a month long data set (see Appendix B). During the test period the receiver was stationary and the "true" position for checking the errors was established as an average of ECEF positions reported by the receiver's internal software. Clock offsets and drifts, which do not stay constant during the experiment, were approximated by fitting a linear function to a sliding window of pseudorange measurements (details are discussed in Section 5.4.1).

4.2.1 Robot State

Unlike position domain integration, this method requires specific way of storing the position and velocity of the robot and additional state data in the samples used in MCL.

First of all, all of the GPS calculations are performed in ECEF coordinate system, which means that it is convenient to store the position and velocity of the robot in this coordinate system. Next, we need to estimate a receiver clock offset together with position, so this is another variable which must exist in the state. To work with velocities, the state variables will also have to contain robot velocities in ECEF reference frame and the receiver clock drift.

The residual errors are modeled as residual clock offset and residual clock drift one pair of variables for each satellite.

4.2.2 Preprocessing

Relatively large amount of work is dedicated to receiver-specific preprocessing of the GPS data. In short, this consists of merging ephemeris to the measurements, removing any inconsistencies in data caused by the receiver and converting the data to sequences of pseudoranges and Doppler measurements. These steps are discussed in Sections 5.1.2, 5.1.3 and 5.1.5.

4.2.3 Motion Model

Since GPS utilizes additional state variables, we need to specify the motion model that governs these variables during the prediction phase of MCL.

Both the receiver clock drift and the residual clock drifts are modeled as a Gaussian random walk. Gaussian distribution was chosen because it provides a good enough approximation.

From the experimental data the following values were calculated:

$$\mathcal{N}(\mu = -8.2 \times 10^{-6}\,\mathrm{m}, \sigma = 0.0400\,\mathrm{m}) \tag{4.6}$$

for the receiver clock drift and

$$\mathcal{N}(\mu = 1.4 \times 10^{-4}\,\mathrm{m}, \sigma = 0.0172\,\mathrm{m}) \tag{4.7}$$

for the residual clock drifts. Clock offsets are obtained by integrating the clock drifts.

Figures 4.3 and 4.4 contain histograms of clock drifts derivation and residual clock drifts derivation.

For comparison, Figures 4.5 and 4.6 show position errors and a histogram of position errors of all the satellite signals together, without corrections of residual errors. In Figure 4.5 each color of the dots corresponds to a single satellite.

4.2.4 Sensor Model

Sensor model of the individual measurements calculates the probability of this measurement being observed, conditioned by the robot state.

Ephemeris

We treat position and velocity of the satellite transmitting the current ranging message as part of the measurement. In reality this information is separate in the transmission, but this approach works without problems when processing the GPS data from the receiver on-line, because the GPS contains current best estimate of the ephemeris periodically repeated in the broadcast navigation data.

In case only the pseudorange measurements were available, we would need to access the ephemeris data from the time of transmission. This might in fact improve quality of the localization if precise ephemeris was used instead (see Section 2.2.6).

Corrections

Pseudoranges and velocities reported by the receiver must be compensated for receiver and satellite clock offset and drift and possibly also for other sources of errors.

For pseudoranges, the clock offsets are compensated according to Equation (2.4). Other errors should be compensated as described in Section 2.4.3 (our implementation uses only tropospheric corrections, see Section 5.4.2.

In a similar way, velocities must be compensated for clock drifts as in (2.9).

Expected Distance and Velocity

To calculate the probability, the algorithm needs to know the distance and relative velocity between the satellite and the potential position of the robot represented by

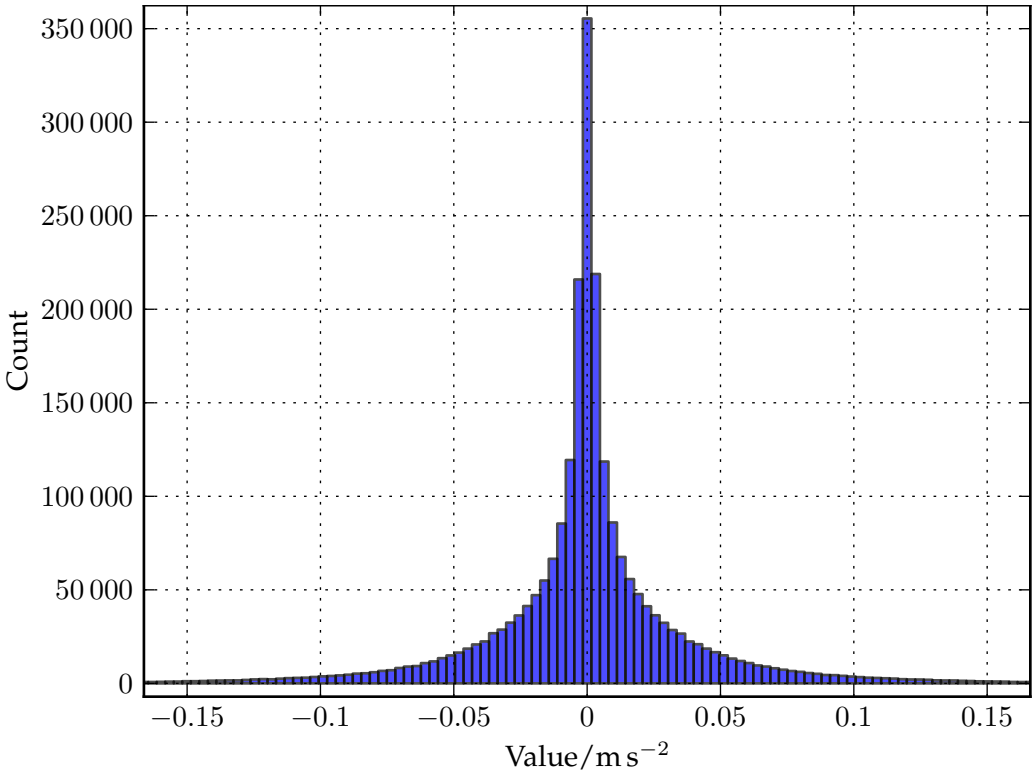


Figure 4.3: Histogram of clock drifts derivation.

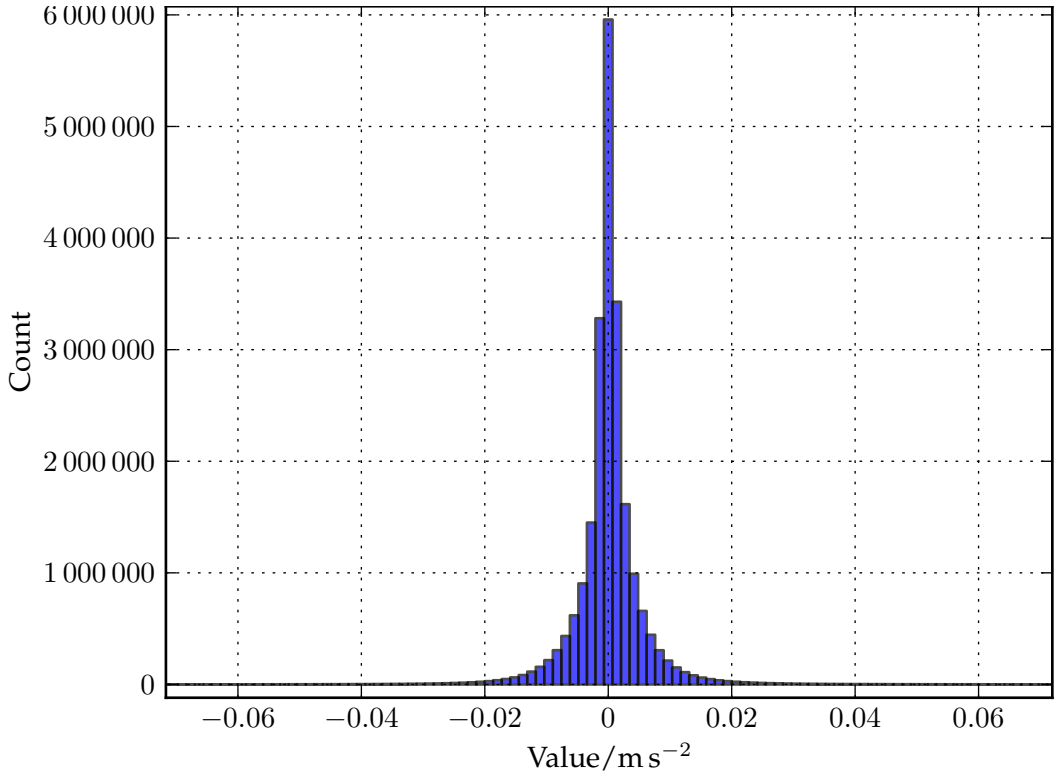


Figure 4.4: Derivation of residual clock drifts.

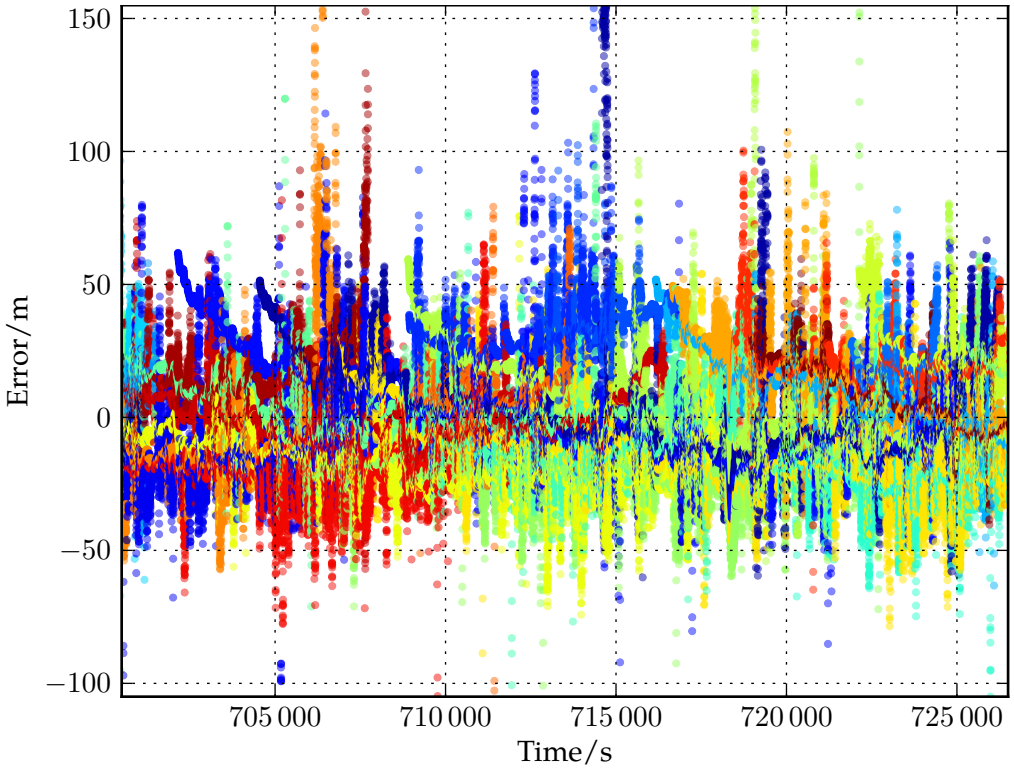


Figure 4.5: Errors for all satellites, without correcting residual errors.

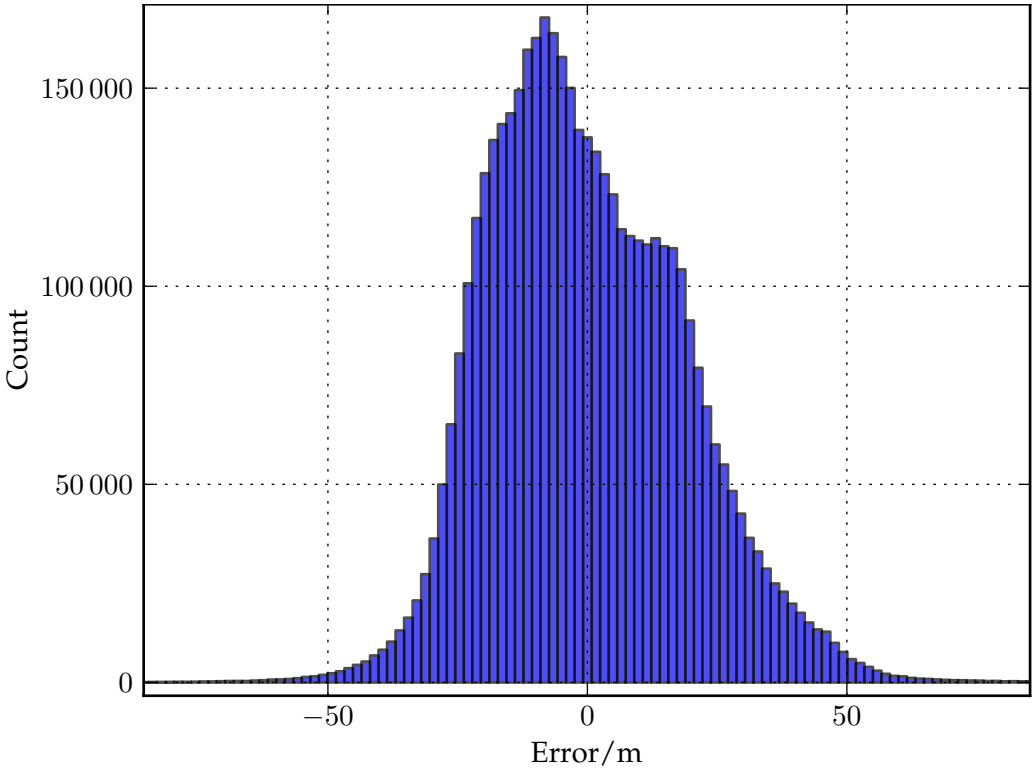


Figure 4.6: Histogram of errors for all satellites, without correcting residual errors.

the sample. Expected distance is simply a length of the vector between the position of the SV and the sample. Expected velocity is calculated according to equation (2.7). Position and velocity errors are then calculated as a difference between the expected and real corrected pseudorange or velocity.

Thresholds

To avoid outliers that frequently appear in the received data (from several hundreds of meters to thousands of kilometers far from the expected measurement), we ignore measurements appearing further than a given threshold from the expected position. The decision whether to throw away a given measurement is made after all available corrections are applied. Values used for the threshold are $150\,\mathrm{m}$ for pseudorange and $4\,\mathrm{m\,s^{-1}}$ for the velocity measurements.

To handle the case of catastrophically wrong estimate in the Monte Carlo localization, thresholded values are assigned probabilities corresponding to the probabilities of a measurement being outside the threshold in the experimental data.

For our datasets these values are 8.42×10^{-3} for pseudoranges and 0.135 for velocities. The surprisingly large probability of velocity measurements outside the threshold is caused by the carrier frequency problem, which is also visible in Figure 4.8 and discussed in Section 5.1.4.

Error Distribution

To obtain probability of a measurement that is within the threshold, we evaluate PDF of normal distribution.

Again, based on the measured data we calculated the parameters of the distributions. For pseudoranges it is

$$\mathcal{N}(\mu = -0.023\,\mathrm{m\,s^{-2}}, \sigma = 7.062\,\mathrm{m\,s^{-2}}) \tag{4.8}$$

and for the velocities

$$\mathcal{N}(\mu = -1.039\,\mathrm{m\,s^{-2}}, \sigma = 1.696\,\mathrm{m\,s^{-2}}) \tag{4.9}$$

Figures 4.7 and 4.8 show the histograms of pseudorange and velocity measurements.

In this case the Gaussian distributions are used because normal distribution of measurement errors is a common assumption in the GPS theory and also because for pseudorange errors this distribution seem to provide good enough estimates.

The odd shape of velocity measurements probability distribution is possibly caused by following an error in the SiRF binary protocol manual, however we didn't manage to find a correct fix for this. This effect is discussed in slightly more detail in Section 5.1.4. Meanwhile, we are using the mentioned normal distribution as a very simple (and slightly incorrect) workaround.

Figure 4.9 contains a plot of pseudorange errors in time. In this plot blocks with larger errors are clearly visible. Detecting these blocks and switching to different error distribution inside them is one of the major improvements to be made in the follow up work.

Figure 4.10 shows a similar plot for velocity errors.

4.2.5 Initialization

The intended use of this algorithm is to improve position estimate when using a consumer grade GPS receiver, so we are not limited by the need to bootstrap the estimates. For localizing the robot globally, we can initialize the position estimate from a fix provided by the receiver's firmware. The initial position estimate can be modeled as discussed in Section 4.1.

Clock offsets and drifts can be initialized from the data provided by the receiver, although there is no standardized way of doing this. As an alternative, clock offsets can be estimated from the positions that were obtained in the previous step. To do this we can assume that the next pseudorange measurement is error-free and obtain clock correction from equation (2.4). Equation (2.9) can be used in a similar fashion to get receiver clock drift.

Since the residual clock drifts and offsets are not necessary for the algorithm, they can be initialized to zero.

4.2.6 Algorithm

Figure 4.11 sums up the measurement domain integration algorithm for MCL, constants appearing in the pseudo code again originate in our measured data.

The motion model is implemented in function `predictGPS()`. It is expected to be called from the function `sampleFromActionModel()` in Figure 3.2, as a part of the prediction step. This function advances the clock drift by a random amount and integrates the current drift to form a new clock offset, both for the receiver clock drift and offset and for the clock drifts and offsets of individual satellites. Function `time()` returns the value of the system performing the localization and is not required to be synchronized to GPS time in any way.

The sensor model in function `observationProbabilityGPS()` processes each measurement and constitutes a core of the measurement domain algorithm. As a first step it calculates effective clock drift as a sum of the receiver clock drift and per-SV residual clock drift. Then an effective clock offset is calculated as a sum of the receiver clock offset and of the residual clock offset. Next a corrected pseudorange and velocity are computed. Here the expression for calculating the velocity is based on a reported velocity as used in the SiRF receiver, instead of the full calculation from the reported frequency. For pseudoranges the delays introduced in the atmosphere must

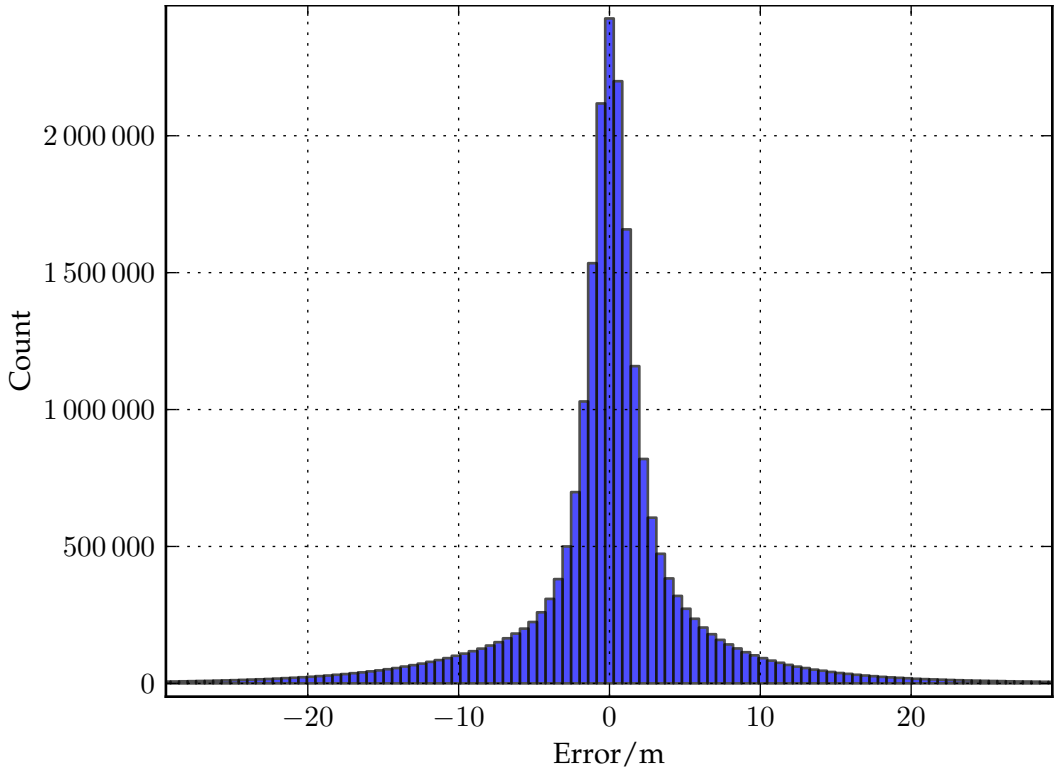


Figure 4.7: Histogram of pseudorange errors.

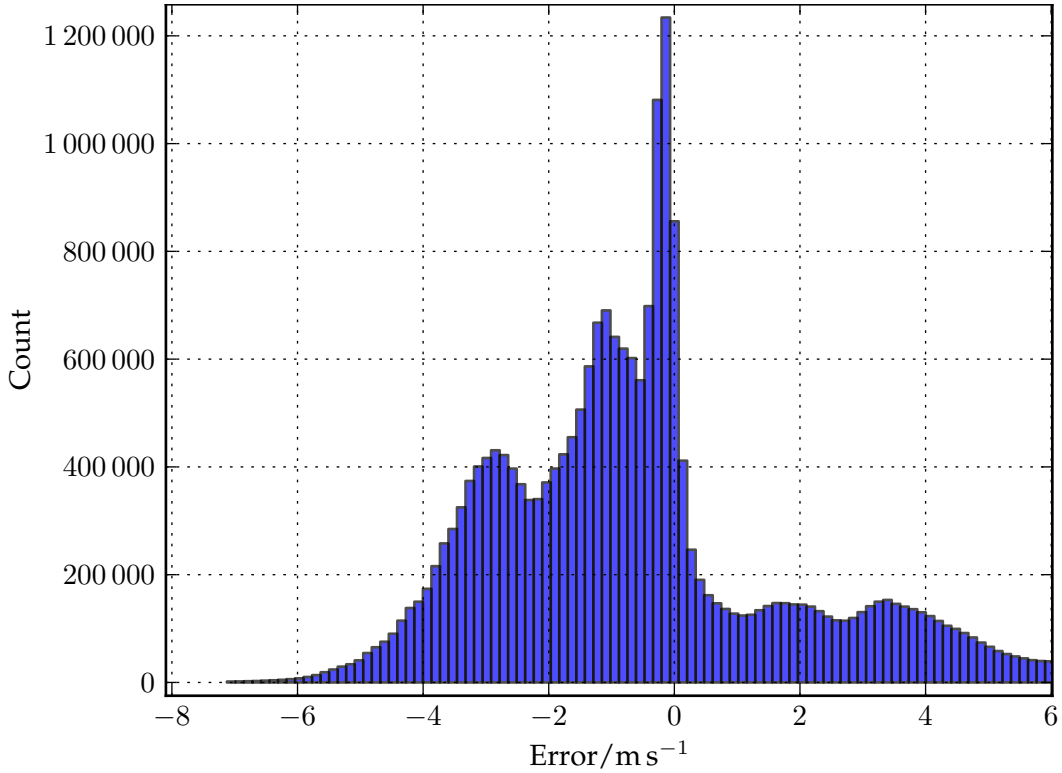


Figure 4.8: Histogram of velocity errors.

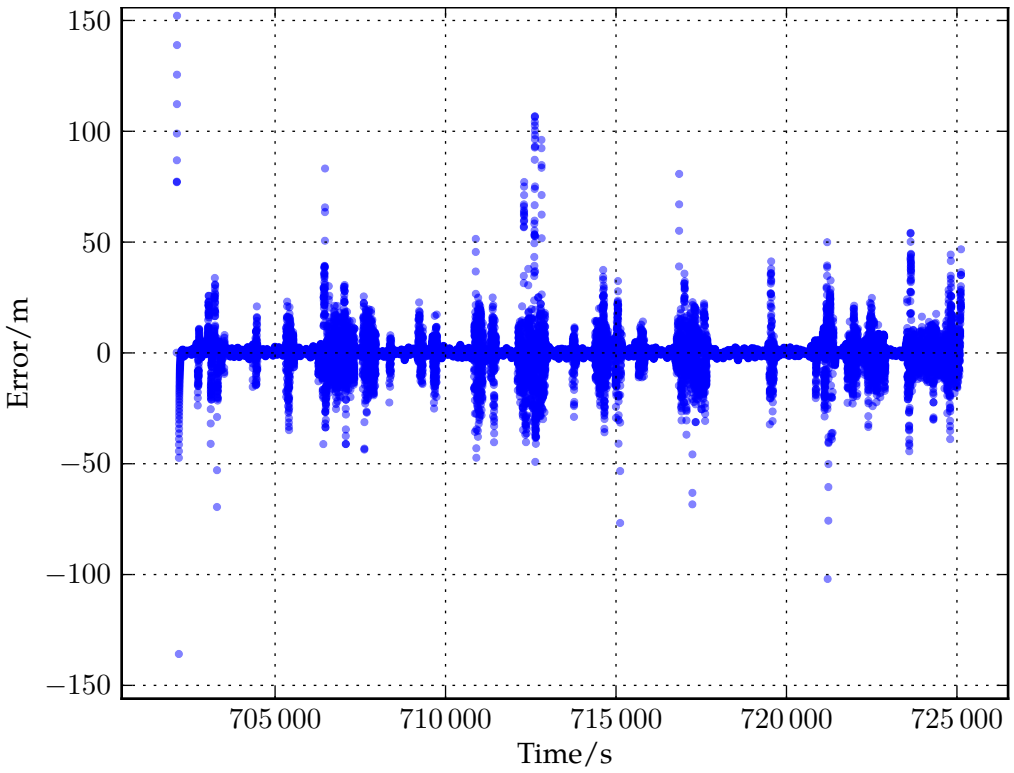


Figure 4.9: Pseudorange measurement errors of satellite 6.

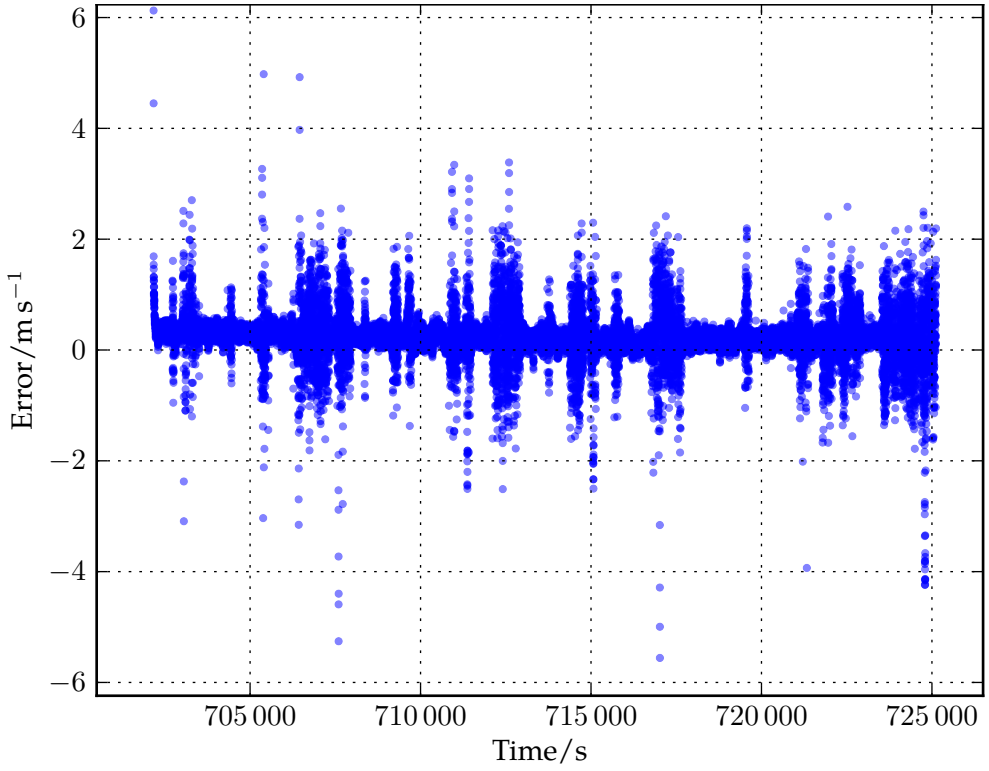


Figure 4.10: Velocity measurement errors of satellite 6.

be compensated, here represented by the function `delays()`. Then the expected range and velocity are calculated. Range computation is trivially the length of the vector between the user and satellite positions in ECEF coordinate frame, expected velocity is obtained as a dot product of unit vector from the user to satellite and velocity distance. Finally the probabilities are acquired from normal distribution PDF, if they are within threshold.

```
function predictGPS :
    Input: list of samples
    Output: modified list of samples

    deltaT ← time() - lastTime
    lastTime ← time()
    foreach s ∈ samples do
        s.clockDrift ← s.clockDrift + norm(μ = −8.2 × 10^−6, σ = 0.0400) ∗ deltaT
        s.clockOffset ← s.clockOffset + s.clockDrift ∗ deltaT

        foreach sv ∈ svList do
            s.svDrift[sv] ← s.svDrift[sv] + norm(μ = 1.4 × 10^−4, σ = 0.0172) ∗ deltaT
            s.svOffset[sv] ← s.svOffset[sv] + s.svDrift[sv] ∗ deltaT
        end
    end
    return samples
end

function observationProbabilityGPS :
    Input: sample, observation
    Output: probability

    sv ← observation.sv
    clockDrift = sample.clockDrift + sample.svDrift[sv]
    clockOffset = sample.clockOffset + sample.svOffset[sv]

    pseudorange ← observation.pseudorange + c ∗ sv.clockOffset − c ∗ clockOffset
        − delays(sv, sample)
    velocity ← observation.velocity − c ∗ clockDrift

    userToSv ← sv.position − sample.position
    relativeVelocity ← sv.velocity − sample.velocity
    geomRange ← abs(userToSv)
    geomVelocity ← dotProduct(relativeVelocity, userToSv) / geomRange

    rangeError ← pseudorange − geomRange
    if abs(rangeError − -0.023) < 150 then
        probability ← normpdf(rangeError, μ = −0.023, σ = 7.062)
    else
        probability ← 8.42 × 10^−3
    end
    velocityError ← velocity − geomVelocity
    if abs(velocityError − -1.039) < 4 then
        probability ← probability ∗ normpdf(velocityError, μ = −1.039, σ = 1.696)
    else
        probability ← probability ∗ 0.135
    end
    return probability
end
```

Figure 4.11: Motion and sensor models for measurement domain integration.

5 Implementation

This chapter describes the implementation, its design and specifics of the GPS receiver used. Tools required to analyze the error models for the GPS in Monte Carlo Localization were implemented with this work, these are available (together with sources of this text) in the git repository at `https://github.com/bluecube/thesis`.

5.1 SiRF III

All experiments in this thesis were performed with a GlobalSat BR-355 serial GPS receiver – a low cost GPS receiver containing a SiRF III chip set, and the following sections describe specifics of working with this hardware.

SiRF III is a fairly high quality consumer receiver found in many "black boxes" on the market. Apart from the standard NMEA, it can be switched to a proprietary binary protocol which provides access to much more of the chip's features. This mode was used exclusively for all our experiments.

There is only a single publicly available piece of documentation concerning this interface [32], but details about workings of the chip are hard to find and must be pieced together from forum posts (especially the thread [12] at GpsPasSion forums) and trial and error.

5.1.1 Protocol

The SiRF protocol consists of input and output binary messages [32, 25]. During regular operation, a group of messages is transmitted approximately every second, containing equivalents of data available in NMEA, ECEF positions of a current fix and many other details. Messages describing positions of satellites and individual measurements can be enabled on demand.

Messages from a GPS (or from a recording, see Section 5.2.2) can be parsed and viewed using the tool `print-all.py`.

5.1.2 Measurements

The SiRF III chip reports each measurement in a separate message, containing time of receipt, ID of the satellite that transmitted the message, pseudorange, carrier frequency and other values. Time of receipt is given as a time of GPS week, relative to the receiver clock. Pseudoranges returned by the SiRF III chip are not corrected for atmospheric delays, but they are smoothed by carrier phase, according to [32]. Raw carrier phase measurements, on the other hand, aren't reported at all.

Measurements reported by the chipset are grouped and appear to have been made simultaneously. Reportedly this is to simplify single point solving of the GPS position.

In later text we will refer to these measurements as groups and use average measurement error of the group in several calculations.

5.1.3 Ephemeris

Instead of providing the Keplerian parameters that are used in the GPS system, SiRF III reports the calculated positions and velocities of the satellites in ECEF reference frame in given time. The times in which the satellite positions are sampled are usually different from the times of measurements, so they have to be interpolated with linear functions before use.

The experiment `previous-sv-state.py` calculates the difference between positions obtained from interpolating satellite positions from two successive linearization points, but in practice these are negligible.

5.1.4 Carrier Frequency

SiRF III reports the equivalent relative velocity between the receiver and satellite in the field named "carrier frequency". According to the manual, it must be compensated for receiver clock drift. Since the reported carrier frequency depends on both receiver clock offset and the satellite clock offset, this means, that the reported carrier frequency must already be corrected for satellite clock drift. Still, this doesn't explain how the carrier frequency should fit to the Doppler effect formula shown in Equation (2.9).

This discrepancy suggests an error in the SiRF manual (an opinion which has already been suggested on the GpsPasSion forum). This would be an explanation for the unexpected probability distribution in Figure 4.8, but we didn't investigate this and meanwhile settled for describing the distribution with a less than optimally fitting normal distribution.

5.1.5 Clock Jumps in SiRF Output

According to notes for message ID 28 in [32], SiRF chipset has a notion of nominal clock drift with corresponding velocity correction of $18\,315.766\,\mathrm{m\,s^{-1}}$. This causes the clock

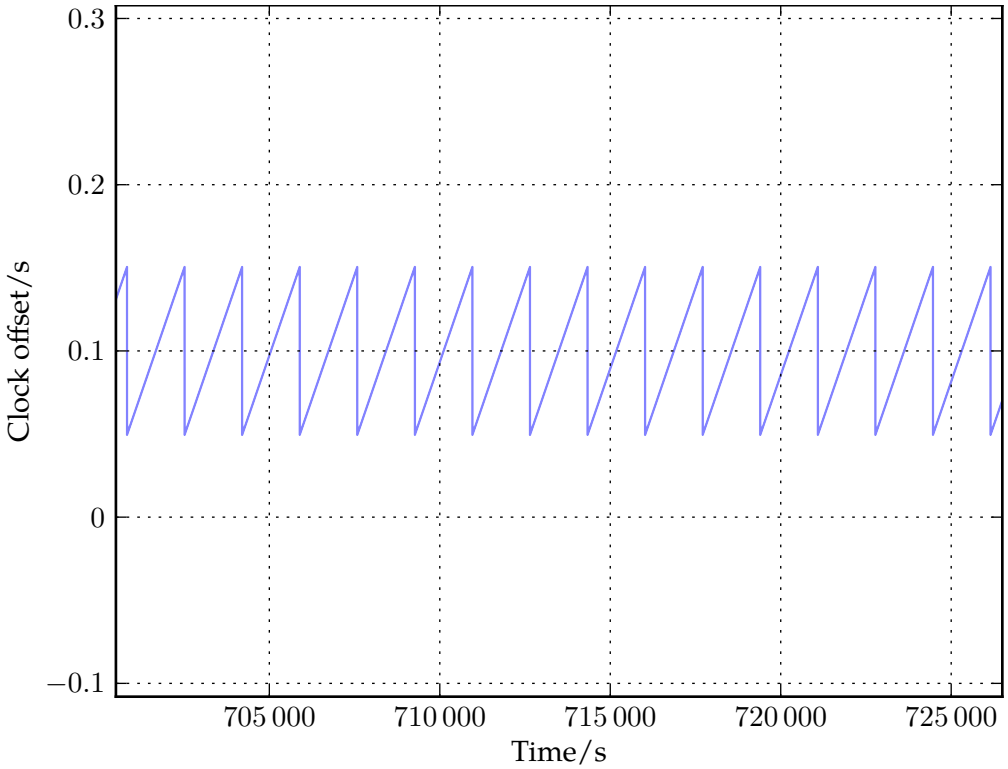


Figure 5.1: Example of clock jumps in SiRF output data

offset to consistently increase in time. When it becomes too large, the chip produces a skip in its reported time and partially corrects the offset. In the recorded data, this happens approximately every 20 min and proves troublesome when processing the raw data, because of the receiver clock not being monotonic. Example of this is shown in Figure 5.1.

This problem is difficult to solve properly without knowing the exact moment and magnitude of clock correction. We work around it by measuring the difference between consecutive average measurement errors and when it becomes very large (larger than 1×10^6 m proves as an appropriate value), we mark this measurement as a skip and estimate the skip length to match the difference between previous pair of average errors.

5.2 Design

Software created for this thesis is designed as a python package handling communication with GPS and several executable scripts. Each of the scripts processes the recorded data in some way and outputs calculated values and graphs seen through this text. Apart from that, the basic idea present through the implementation is to record all data in advance and process these recordings offline.

Python programming language was chosen for the implementation, because of the speed of development and large number of available libraries. This code is intended only as a proof of concept and for further experimenting and it does have performance problems when running on large batch data. A "real world" implementation would probably be written in C, C++, or another compiled language for increased performance and better portability to embedded systems, using libraries like GPSTk [37] for the low level GPS access.

For running the python code, python 2.7 is required. The following packages are used in addition to the standard library:

- matplotlib
- numpy and scipy
- progressbar
- pyproj
- pyserial is required only if interfacing with a real GPS receiver is intended. It is not necessary for working only with recordings.

5.2.1 Communication With SiRF Chip

The most basic function of the `GPS` package is serial communication with the SiRF chip.

Serial Communication and Protocol Detection

Operations with serial port are performed by the pySerial library. Because the chip can work in several modes of communication, it is necessary to switch it to a known mode before use. This is done by attempting to read messages from the GPS in two expected modes (4800 8N1 NMEA protocol and 115200 8N1 SiRF binary protocol). If neither of these fits, all other possible combinations of baud rate and protocol are tried.

Once the GPS mode is detected, it is switched to 115200 8N1 SiRF binary. The rest of operations with GPS are performed in this mode. NMEA messages can be parsed, but only the ones necessary for switching to SiRF binary protocol are implemented.

Processing the Protocol

Messages of the SiRF binary protocol are parsed from the input stream and representing objects are created based on classes in the file `gps/sirf_messages.py`. The message parsing code uses introspection in python to enumerate message types available for parsing.

5.2.2 Recordings

Recordings of GPS messages are stored in a custom format designed to drop store as much information about the original stream as possible. It consists of a sequence of

timestamped binary SiRF messages, the whole stream compressed with gzip.

A custom file format was chosen instead of one directly available in python, because it allows us to store data incrementally, without keeping all of the messages in memory. Even though parsing of the recordings is faster than when this was attempted using standard pickle file format, it is still optimized for simplicity and admittedly creates a performance bottleneck.

Recording files are intended to be seamlessly interchangeable with a real GPS device, which proved to be very useful mainly in the early phases of experimentation.

The script `record.py` stores data from a real serial GPS (or from other recording), the script `checksum.py` calculates CRC32 checksums of the stored data.

To save time on higher level experiments, some of them (for example the error tests in files `errors-*.py`) use pre-calculated data saved in numpy data dump format. Data in this format don't have to be parsed on every load and also some basic preprocessing steps are not needlessly repeated. The preprocessed data files are created using the script `wgs84_fixes_to_numpy.py` for WGS84 positions of fixes and the script `clock_offsets_to_numpy.py` for pseudorange and velocity data.

5.2.3 Interfaces to GPS Data

There are several levels of access to the data from GPS or replay available, ranging from returning individual binary strings of unparsed data to an option based on observers that are notified every time a desired type of message is encountered.

5.2.4 Ephemeris

Since SiRF III sends complete calculated positions of satellites in its binary protocol, we are using these, although it is possible to switch to a different ephemeris source relatively easily. An alternative ephemeris source, which provides precise ephemeris data from the IGS service [34] is implemented, although it is currently unused.

Both of these are based on the previously mentioned observer framework for processing received GPS messages.

5.3 Position Domain Error Model

Calculating error model in position domain is done in a fairly straightforward way. The obtained WGS84 coordinates are projected onto a two dimensional plane (details are discussed later). The fix data are not obtained from a NMEA model, but from message 41 in SiRF binary protocol. This lets us simply work with both WGS84 and pseudorange on the same recording, while keeping the available values identical to what's found in NMEA protocol.

Next, because the receiver was stationary during the whole recording, we want to calculate the ground truth position. To do this we assume that the reported point position errors are zero mean. This is a reasonable assumption for long recordings, because the distribution of satellites in view during a longer period is virtually uniform which makes any potential biases in pseudorange measurements cancel out. This means we can obtain approximation of the real receiver position as a simple mean of reported GPS positions. Then for each measurement a distance from the mean position is calculated and curves for each model are fitted.

5.3.1 Mapping Coordinate Frames

Conversion between WGS84 latitude and longitude and two-dimensional euclidean plane necessary for localization is done using orthographic projection from the library PyProj [26].

The exact choice of projection is not that important for robots operating within a radius of a few kilometers, but orthographic projection was chosen because it's principle is simple to imagine.

5.4 Measurement Domain Error Model

The basic idea for determining error model for pseudorange measurements is similar to how error model for position domain is determined, but it is complicated by several aspects of the pseudorange measurements.

In contrast with the previous approach, the whole localization process is kept in the ECEF coordinate frame, because of the amount of GPS data being processed.

The main problem is determining receiver position from which the errors will be calculated. Spatial position can be determined fairly simply, since SiRF receiver also reports position in ECEF reference frame and averaging these values gives reasonable estimate, for the same reasons as averaging could have been used in position domain integration.

5.4.1 Clock Offset Estimation

Clock offset of the receiver, on the other hand, is not fixed during the recording and has to be estimated during the calculation. Since we know geometric component of each pseudorange measurement, we can calculate the receiver clock offset under the condition that there is no measurement error.

The previous step gives us a large number of points, one for each pseudorange measurement, with time and the idealized receiver clock offset. It is now necessary to merge all the clock offsets to obtain clock offset for each measurement time.

First approach explored was to fit a polynomial through these points, but it didn't fit the points well enough on large data sets. Currently we fit a linear function to points from a sliding window centered around each of points as seen on Figure 5.2.

An input of this procedure is a set of points $\{x_i, y_i\}$. The value x_i meaning time of the measurement and y_i a clock offset of individual measurements. For each i we select points $P_i = \{(x_j, y_j) : |x_j - x_i| < \frac{w}{2}\}$, where w is a width of the sliding window. The coefficients a_i and b_i of linear functions $a_i(x - x_i) + b_i$ fitted through the points P_i using least squares are used as a clock drift and clock offset of the dataset. This calculation can be performed in $\mathcal{O}(n)$ time, which is very important for the sizes of datasets we deal with in this work.

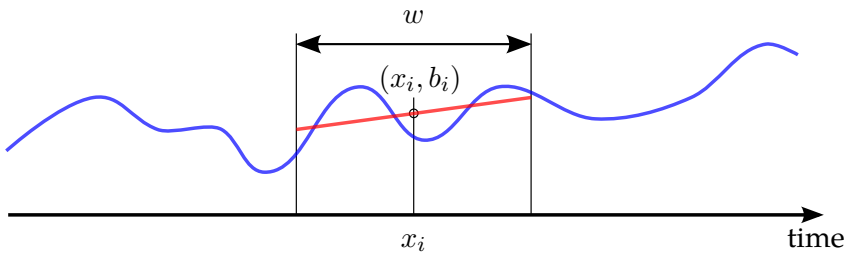


Figure 5.2: Sliding window linear regression.

The clock offset fitting step is done twice in our code, first run done on all data, then the errors are calculated for the first time and the second run is performed only on measurements with error lower than a certain threshold.

Empirically obtained values are 10 min for the window width and 150 m for outlier threshold. In this case we don't have to deal with problems of extremely large errors in estimation like in Section 4.2.4.

A possible objection to this reasoning is, that this way the estimated receiver clock offset will also include other effects from different sources than receiver clock, for example atmospheric delays (imagine the user standing in a ball of material that slows down propagation of radio signals), or signal processing delays in the receiver. This is true, but as far as localization is concerned, these effect would be indistinguishable from receiver clock inaccuracies and will be removed also during localization in a similar way.

5.4.2 Applied Corrections

The web site [17] was very useful when implementing corrections for pseudorange measurements.

We to implemented corrections for ionospheric and tropospheric effects:

- Tropospheric corrections, implemented based on the Hopfield's tropospheric model with the mapping function implementation based on [17], improved the

one sigma error by approximately 0.2 m.

- Ionospheric corrections taken from the values transmitted in the navigation messages (and accessed in the SiRF SV state messages) on the other hand, increased the one sigma error by 0.05 m. Because of this we chose not to use the ionospheric correction. This effect may be caused by our implementation estimating the residual clock offsets – the ionospheric effects are already corrected and the inaccurate estimation transmitted by the space segment only adds noise to our data.

6 Conclusion

The goal of this thesis was to explore methods of utilizing of a low cost GPS receiver for outdoor robot localization in the framework of the Monte Carlo localization algorithm. Two approaches to perform this task were selected.

Firstly, the "obvious" method of interfacing to the GPS receiver with the standard NMEA protocol was evaluated. According to this approach the complete fixes in WGS84 coordinates were used, with latitude, longitude and HDOP being the only values used in the localization. Error model was constructed from a set of experimentally measured data, giving one sigma precision of approximately 7 m.

Secondly, a new method was developed, working with GPS one level of abstraction deeper. This method makes use of the individual pseudorange and velocity measurements and compares the reported and expected distances and relative velocities between the receiver and the GPS satellites. To obtain the raw measurements for this method, communication with the GPS receiver had to be performed using the proprietary binary protocol, since such detailed data are not available in the standard NMEA messages. Again, an error model was calculated from recorded data. In the current state, this method offers a precision comparable to the simple approach, showing room for improvement in follow-up work. This provides position estimates free of the assumptions made in the NMEA data and offers superior error characterization.

To construct the error models and to visualize the data, a set of tools was implemented. These consist of a library for communication with SiRF III GPS receivers and a group of individual scripts for experimenting with various aspects of the GPS signals. During development of these tools we successfully overcame a number of problems caused mainly by the lack of documentation of the SiRF III chip operation details.

In the current state, this work could be of benefit to the designers of outdoor robots, allowing them to simply employ the first method for modeling the error of GPS in NMEA mode. Our second method provides the robot designers with comparable precision, velocity processing, more detailed characteristics of the position error and a framework facilitating insight into the localization process and further precision improvements. On top of that we provide a framework for experiments with SiRF receivers for anyone interested. We also hope this text might serve as another piece of the fragmented information about the SiRF III chipset on the Internet.

Bibliography

[1] European Space Agency. *What Is Galileo?* URL: http://www.esa.int/esaNA/GGGMX650NDC_galileo_0.html.

[2] Sanjeev Arulampalam et al. "A Tutorial on Particle Filters for On-line Non-linear/Non-Gaussian Bayesian Tracking". In: *IEEE Transactions on Signal Processing* 50 (2001), pp. 174–188.

[3] A. Bourdeau, M. Sahmoudi, and J. Y Tourneret. "Constructive use of GNSS NLOS-multipath: Augmenting the navigation Kalman filter with a 3D model of the environment". In: *Information Fusion (FUSION), 2012 15th International Conference on*. 2012, pp. 2271–2276.

[4] Frank Dellaert et al. "Monte Carlo Localization for Mobile Robots". In: *IEEE International Conference on Robotics and Automation (ICRA99)*. Apr. 1999.

[5] Dale DePriest. *NMEA data*. URL: http://www.gpsinformation.org/dale/nmea.htm.

[6] Julien Diard, Pierre Bessiere, and Emmanuel Mazer. "A Survey of Probabilistic Models Using the Bayesian Programming Methodology as a Unifying Framework". English. In: *International Conference on Computational Intelligence, Robotics and Autonomous Systems (IEEE-CIRAS), Singapore*. France, 2003, p. x.

[7] Vyacheslav Dvorkin and Sergey Karutin. *GLONASS: Current status and perspectives*.

[8] Dieter Fox. "Adapting the Sample Size in Particle Filters Through KLD-Sampling". In: *The International Journal of Robotics Research* 22.12 (2003), pp. 985–1003.

[9] Dieter Fox. "Markov Localization: A Probabilistic Framework for Mobile Robot Localization and Navigation". PhD thesis. Germany: University of Bonn, 1998.

[10] Dieter Fox et al. "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots". In: *PROC. OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE (AAAI)*. 1999, pp. 343–349.

[11] P. Fyfe, K. Kovach, and ARINC Research Corporation. *ICD-GPS-200B-PR Navstar GPS Space Segment/navigation User Interfaces (public Release Version)*. ARINC Research Corporation, 1992. URL: http://books.google.cz/books?id=KiBPtwAACAAJ.

[12] *GpsPasSion Forums — Sirf nav data physical model*. URL: http://www.gpspassion.com/forumsen/topic.asp?TOPIC_ID=54336.

[13] Werner Gurtner and Lou Estey. *RINEX: The Receiver Independent Exchange Format*. 2007.

[14] R.E. Kalman. "A New Approach to Linear Filtering and Prediction Problems". In: *Transactions of the ASME–Journal of Basic Engineering* 82.Series D (1960), 35–45.

[15] E.D. Kaplan and C.J. Hegarty. *Understanding GPS: Principles And Applications*. Artech House Mobile Communications Series. Artech House, 2006. ISBN: 9781580538947.

[16] Jan Kouba. *A Guide to Using International GNSS Service (IGS) Products*. URL: http://igscb.jpl.nasa.gov/igscb/resource/pubs/UsingIGSProductsVer21.pdf.

[17] Sam Storm van Leeuwen. *Sam's GPS Software Pages*. URL: http://home-2.worldonline.nl/~samsvl/software.htm.

[18] J. L. Leva. "An alternative closed-form solution to the GPS pseudo-range equations". In: *Aerospace and Electronic Systems, IEEE Transactions on* 32.4 (1996), pp. 1430–1439. DOI: 10.1109/7.543864.

[19] National Coordination Office for Space-Based Positioning, Navigation, and Timing. *GPS.gov: New Civil Signals*. URL: http://www.gps.gov/systems/gps/modernization/civilsignals.

[20] National Imagery and Mapping Agency. *Department of Defense World Geodetic System 1984: its definition and relationships with local geodetic systems*. Tech. rep. TR8350.2. St. Louis, MO, USA: National Imagery and Mapping Agency, June 2004.

[21] Navipedia. *Klobuchar Ionospheric Model — Navipedia*, 2012. URL: http://www.navipedia.net/index.php?title=Klobuchar_Ionospheric_Model&oldid=11615.

[22] Navipedia. *NeQuick Ionospheric Model — Navipedia*, 2012. URL: http://www.navipedia.net/index.php?title=NeQuick_Ionospheric_Model&oldid=11618.

[23] Navipedia. *Tropospheric Delay — Navipedia*, 2013. URL: http://www.navipedia.net/index.php?title=Tropospheric_Delay&oldid=12152.

[24] David Obdržálek. "Robot Localization". PhD thesis. Charles University in Prague, Faculty of Mathematics and Physics, 2012.

[25] Bartłomiej Oszczak and Krzysztof Serżysko. "Decoding of SiRF Binary Protocol". In: *Artificial Satellites* 46 (Apr. 2012), pp. 127–137.

[26] *pyproj: Python interface to PROJ.4 library*. URL: http://code.google.com/p/pyproj/.

[27] Chris Rizos. *Principles and Practice of GPS Surveying*. 1999. URL: http://www.gmat.unsw.edu.au/snap/gps/gps_survey/principles_gps.htm.

[28] *Roboauto*. URL: http://www.roboauto.cz/foswiki/bin/view/Projekt/RoboautoAbout.

[29] *Robotour*. URL: http://robotika.cz/competitions/robotour/en.

[30] Mark Schenewerk. "A brief review of basic GPS orbit interpolation strategies". In: *GPS Solutions* 6 (4 2003), pp. 265–267. ISSN: 1080-5370.

[31] Jun Shen. *COMPASS/Beidou – China's GNSS*. 2009.

[32] SiRF Technology, Inc. *SiRF Binary Protocol Reference Manual*.

[33] A. F. M. Smith and A. E. Gelfand. "Bayesian Statistics without Tears: A Sampling-Resampling Perspective". In: *The American Statistician* 46.2 (1992), pp. 84–88.

[34] National Geodetic Survey. *GPS Orbits*. URL: http://www.ngs.noaa.gov/orbits/orbit_data.shtml.

[35] Sebastian Thrun et al. "Robust Monte Carlo localization for mobile robots". In: *Artificial Intelligence* 128.1–2 (2001), pp. 99 –141. ISSN: 0004-3702.

[36] Nils Ole Tippenhauer et al. "On the requirements for successful GPS spoofing attacks". In: *Proceedings of the 18th ACM conference on Computer and communications security*. CCS '11. New York, NY, USA: ACM, 2011. ISBN: 978-1-4503-0948-6.

[37] Brian Tolman et al. "The GPS Toolkit: Open Source GPS Software". In: *Proceedings of the 16th International Technical Meeting of the Satellite Division of the Institute of Navigation*. Long Beach, California, Sept. 2004.

[38] N. Viandier et al. "GNSS Performance Enhancement in Urban Environment Based on Pseudo-range Error Model". In: *PLANS - Position Location and Navigation IEEE Symposium*. 2008, pp. 1079–1089.

[39] Greg Welch and Gary Bishop. *An Introduction to the Kalman Filter*. Tech. rep. Chapel Hill, NC, USA, 1995.

[40] David L. Wilson. *HDOP and GPS Horizontal Position Errors*. 2010. URL: http://web.archive.org/web/20100908132322/http://users.erols.com/dlwilson/gpshdop.htm.

A DVD Contents

- `/` — Root of the DVD
  - `impl/` — Implementation
  - `data/` — Recorded data
    - `preprocessed/` — Preprocessed recorded data (see Section 5.2.2)
  - `repository/` — Snapshot of the git repository
  - `thesis.pdf` — This text
  - `usage.txt` — Brief usage information for scripts in `/impl/`

B Datasets

Three sets of measured GPS data were recorded for this work. All of them were recorded using GlobalSat BR-355 serial GPS receiver with SiRF III chip in the SiRF binary protocol.

- First dataset, contains approximately one month of recorded SiRF data. During this period the receiver was stationary inside a room and had a good signal reception. It was used for modeling of the GPS measurements errors in Chapter 4.

  The dataset contains $2\,363\,458$ measurement cycles and $22\,726\,200$ pseudorange measurements.

  For quick testing, we also keep a subset of this recording containing the first night of the dataset and of first week.

  On the DVD the main recording of dataset is located in `/data/month.recording2`, the shortened versions can be found in `/data/one_night.recording2` and in `/data/one_week.recording2`.

- The second dataset consists of two recordings made during the Robotour competition [29] on a Roboauto Karlík [28] robot.

  In total the dataset contains 1531 GPS measurement cycles with $16\,261$ pseudorange measurements, accompanied by sensor recordings from Karlík.

  Recordings for this dataset can be found in files `/data/roboauto1.recording2` and `/data/roboauto2.recording2`. Karlík's log files are located in the directory `/data/roboauto/`.

- The last dataset contains approximately 30 minutes recorded with stationary receiver with poor signal quality.

  The recording is in the file `/data/30_minutes_weak_signal.recording2`.

Figure B.1 contains histograms showing the distribution of HDOP values in the datasets.

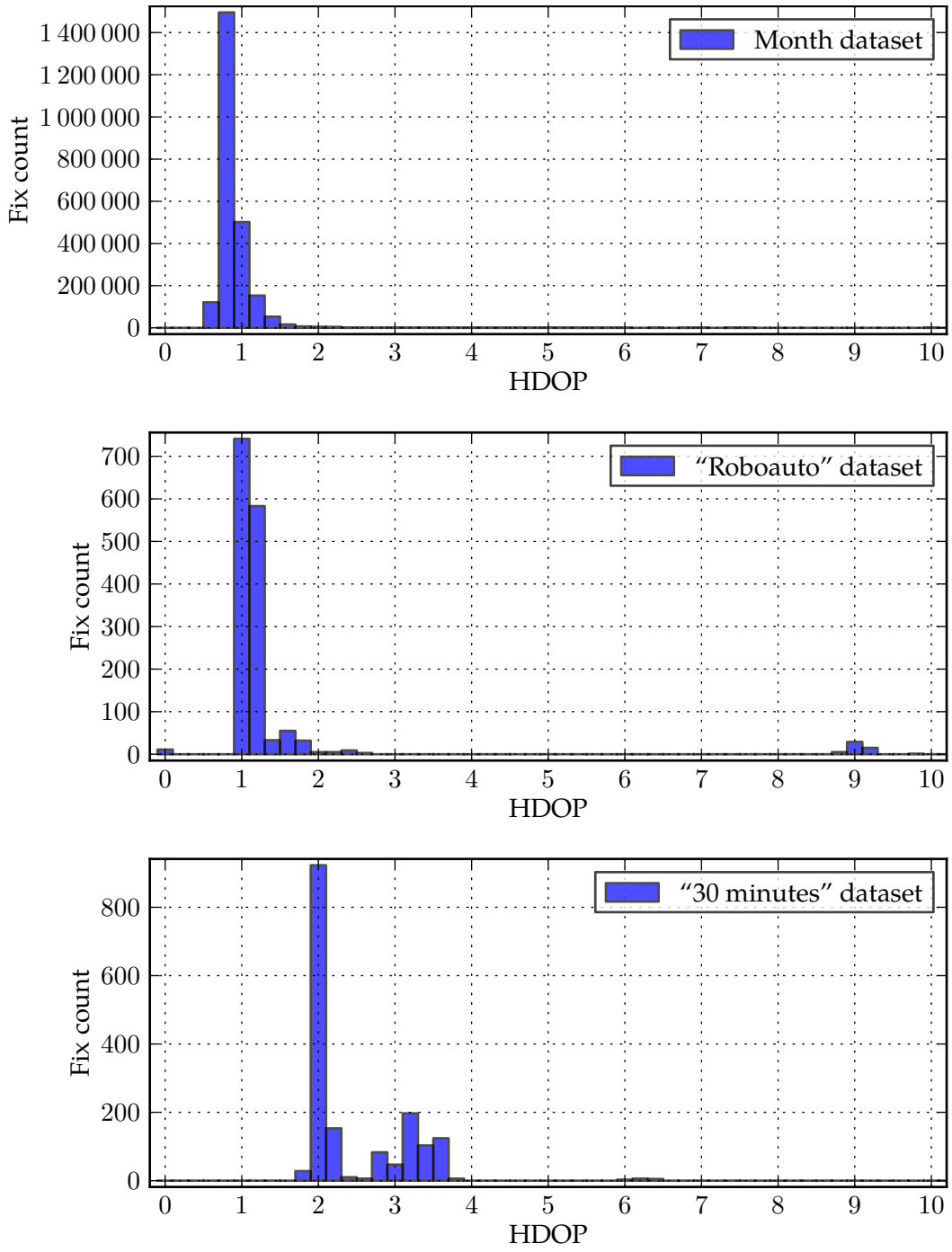


Figure B.1: Histograms of HDOP values for our data sets.

C tl;dr

This chapter provides an overview of the thesis in a form as condensed as possible. Intended for anyone who knows both GPS and MCL and dislikes reading long texts.

- Using low level GPS data as a sequence of correction inputs to MCL.
- Communication with GPS receiver
  - Standard NMEA protocol doesn't contain enough detail for advanced usage.
  - Support for SiRF III chips with binary SiRF protocol is implemented, but the approach should be adaptable to other receivers without major modifications.
  - Section 5.1.
- NMEA data as MCL input
  - Simpler, less CPU intensive, drops some information when the results are wrapped in lat / lon / hdop for WGS84.
  - Working in 2D, no need to modify MCL samples.
  - Fitting curve to measured data:

$$\mathrm{DRMS}(\mathrm{HDOP}) = \sqrt{(4.941\mathrm{HDOP})^2 + 3.568^2}$$

  - Sensor model based on Rayleigh distribution.
  - PDF:

$$f(x) = \frac{x}{\sigma^2} e^{-x^2/2\sigma^2} = \frac{2x}{\mathrm{DRMS}(\mathrm{HDOP})^2} e^{-x^2/\mathrm{DRMS}(\mathrm{HDOP})^2}$$

  - Approximately 7 m DRMS error.
  - Figures 4.1 and 4.2 and section 4.1.2.
- Pseudoranges as MCL input
  - Experimental, more complex, higher CPU usage.
  - Working in ECEF coordinates, tracking clock offsets and residual errors.
  - Needs modifications to MCL sample.
  - Residual errors:
    - Per satellite.
    - Characterized as residual clock drift and residual clock offset during processing individual measurements.

    - Residual clock drift and offset are added to receiver clock drift and offset.
    - Covers ionospheric and other low-frequency errors.
- Motion model:
    - Updating clock drifts, clock offsets, residual drifts.
    - Clock drifts modeled as Gaussian random walk.
        - Receiver clock drifts: $\mathcal{N}(\mu = -8.2 \times 10^{-6}\,\mathrm{m\,s^{-2}}, \sigma = 0.0400\,\mathrm{m\,s^{-2}})$
        - Residual clock drifts: $\mathcal{N}(\mu = 1.4 \times 10^{-4}\,\mathrm{m\,s^{-2}}, \sigma = 0.0172\,\mathrm{m\,s^{-2}})$
    - Clock offsets obtained by integrating clock drifts.
    - Figures 4.3 and 4.4
- Sensor model:
    - Range error:
        - $\mathrm{rangeError} = \mathrm{correctedPseudorange} - \mathrm{geometricRange}$
        - Correcting pseudorange for receiver clock offset + residual clock offset, satellite clock offset, tropospheric errors.
        - Ionospheric errors not corrected explicitly (see Section 5.4.2).
    - Velocity error:
        - $\mathrm{velocityError} = \mathrm{correctedVelocity} - \mathrm{velocityDifference}$
        - Correcting velocity for receiver clock drift + residual clock drift.
        - Encountered a bug in SiRF III (see Section 5.1.4).
    - Thresholding
        - Not using the PDF model for very large errors, instead using fixed probability of exceeding threshold.
        - Threshold $150\,\mathrm{m}$, probability 8.42×10^{-3} for pseudorange.
        - Threshold $4\,\mathrm{m\,s^{-1}}$, probability 0.135 for velocity measurements.
    - Probabilities for measurements within the thresholds:
        - Using PDF of normal distribution.
        - Range errors: $\mathcal{N}(\mu = -0.023\,\mathrm{m}, \sigma = 7.062\,\mathrm{m})$
        - Velocity errors: $\mathcal{N}(\mu = -1.039\,\mathrm{m\,s^{-1}}, \sigma = 1.696\,\mathrm{m\,s^{-1}})$
        - Better fitting probability distribution, or switching probability distributions are possible improvements.
        - Figures 4.7 to 4.9
- Similar precision to the simple model so far, possible improvements.
- Figure 4.11 and section 4.2.